AD-A064 065 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 15/5
OPPORTUNE RESCHEDULING SYSTEM FOR MILITARY AIRLIFT COMMAND CARG--ETC(U)
DEC 78 G F SANDERSON

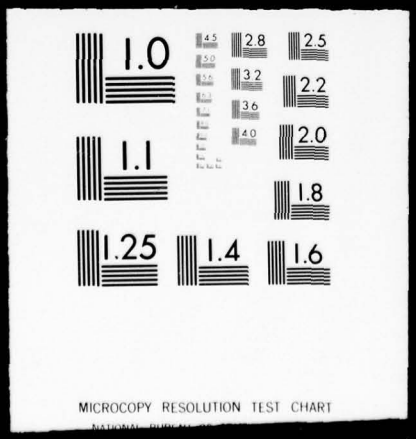
UNCLASSIFIED AFIT/GCS/EE/78-17 NL

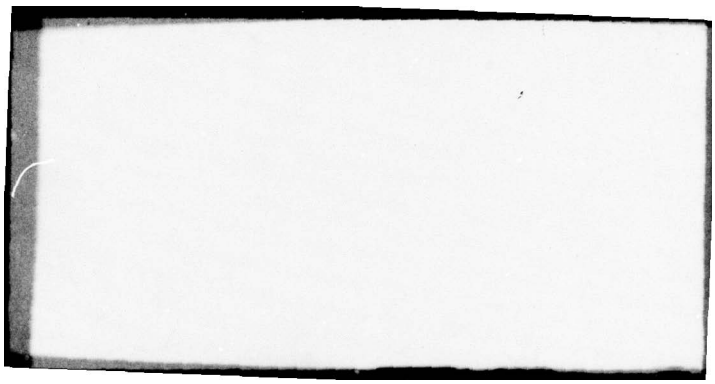
1 OF 3
AD
A064065

1 OF 3

AD

A064065





AFIT/GCS/EE/78-17

LEVEL

AD A064065

DDC FILE COPY.

OPPORTUNE RESCHEDULING SYSTEM FOR MILITARY AIRLIFT COMMAND CARGO AND PASSENGER MISSIONS

THESIS

AFIT/GCS/EE/78-17

Gary F. Sanderson
Major USAF

Approved for Public Release; Distribution Unlimited

(14) AFIT/GCS/EE/78-17

(6) OPPORTUNE RESCHEDULING SYSTEM FOR MILITARY AIRLIFT COMMAND CARGO AND PASSENGER MISSIONS

THESIS

(9) Master's thesis

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

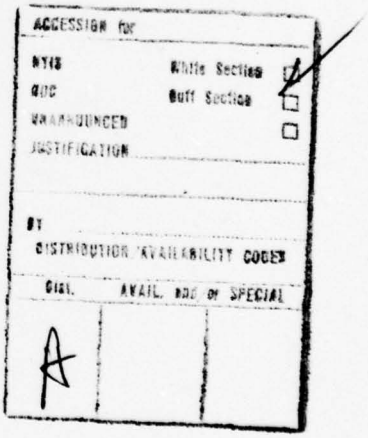
Air University

in Partial Fulfillment of the

Requirements of the Degree of

Master of Science

(12) 219 p.



by

(10) Gary F. Sanderson, ~~B.S.~~, M.A.
~~Major~~ USAF

Graduate Computer Science

(11) ~~December~~ 1978

Approved for Public Release; Distribution Unlimited

012 225

xlt

PREFACE

Col Norman S. Cole was the first to study and document the potential savings which could be realized from rescheduling partially-loaded military cargo and passenger airlift missions to pick up and deliver additional Department of Defense equipment and personnel needing transport. To take advantage of this potential, a branch within the Airlift Operations Directorate of the Military Airlift Command, under the direction of Lt Col Gerald D. Hunter, was assigned the responsibility of matching specific missions and additional airlift requirements. The objective of this thesis was to design and implement a system for providing automated assistance to personnel involved in this rescheduling effort.

Lt Col Hunter and Technical Sergeants Kent Trumpeter and Jim Tawyea provided invaluable assistance in defining a realistic set of system requirements, and in the actual software implementation and testing. Capt Pete Miller contributed expert advice regarding system design, and Major Al Ross was instrumental in the development of the overall organization of the thesis document. Dr Thomas Hartrum provided both technical advice on the numerous problems encountered in the design and implementation, and the overall perspective and guidance to keep the forest visible through the trees. Mrs Eve Vaught was extremely able in converting a handwritten draft into a final, professional-looking product.

Maj Gary Sanderson

TABLE OF CONTENTS

		Page
PREFACE		ii
LIST OF FIGURES		v
LIST OF TABLES		vi
ABSTRACT		vii
I.	INTRODUCTION	1
	Military Airlift Command Opportune Rescheduling	1
	The Objectives of the Management Analysis	3
	The Objectives of the System Design	4
	Overview of the Thesis	5
II.	REQUIREMENTS DEFINITION	6
	Structured Analysis as a Requirements Definition Language	6
	Structured Analysis Activity Model	7
	System Purpose	8
	Manage Airlift Operations (Node A-2)	11
	Plan and Direct Airlift Operations (A-1)	13
	Develop Mission Rescheduling Options/Costs/Displays (A-0)	16
	Select Diversion Eligible Missions (A1)	19
	Pick Most Efficient Missions for Rescheduling (A2)	20
	Print/Plot Selected Data (A3)	22
III.	SYSTEM DESIGN	24
	Primary Design Methodology	24
	Additional Design Considerations	25
	The Basic Design	26
IV.	THE DATA TRANSFER PROGRAM	29
	Data Communications Conventions	30
	Teletype Emulation	30
	The Program	31
V.	THE OPPORTUNE RESCHEDULING SYSTEM PROGRAM	34
	Overlay Structure	35
	The Executive Control Program	37

TABLE OF CONTENTS (cont'd)

Page

V. (Continued)

The MAP PLOT Overlay 37
The ENTER Overlay 42
The DISPLAY Overlay 43
The OPTIMIZE Overlay 44
System Hierarchical Modular Structure 46
Program and Data Storage 56
Data File Organization 57

VI. IMPLEMENTATION RESULTS AND RECOMMENDATIONS 61

Coding and Testing 61
Recommendations for System Improvements 63

BIBLIOGRAPHY 67

APPENDIX A: Structured Analysis Diagrams 69

APPENDIX B: Module Relatedness Measurement 77

APPENDIX C: Hewlett-Packard Language Notation 82

APPENDIX D: WWMCCS Program to Select Diversion Eligible Missions . . 86

APPENDIX E: Description and HPL Code for Transfer Program 89

APPENDIX F: Description and HPL Code for the Executive Control Program 96

APPENDIX G: Description and HPL Code for the MAP PLOT Overlay 113

APPENDIX H: Description and HPL Code for the ENTER Overlay 125

APPENDIX I: Description and HPL Code for the DISPLAY Overlay 139

APPENDIX J: Description and HPL Code for the OPTIMIZE Overlay 153

APPENDIX K: Map Projection Formulas 167

APPENDIX L: Data Record Formats 171

APPENDIX M: User's Guide 178

APPENDIX N: Sample Data Displays 203

VITA 210

LIST OF FIGURES

Figure		Page
2-1	Index to Structured Analysis Diagrams	10
2-2	Manage Airlift Operations (A-2)	12
2-3	Plan and Direct Airlift Operations (A-1)	14
2-4	Develop Mission Rescheduling Options/Costs/Displays (A-0)	15
2-5	Develop Mission Rescheduling Options/Costs/Displays (AO)	17
2-6	Select Diversion Eligible Missions (A1)	19
2-7	Pick Most Efficient Missions for Rescheduling (A2)	21
2-8	Print/Plot Selected Data	23
3-1	Basic System Structure	28
A-1	Top-Down View of an SA Model	72
A-2	Arrow Definitions	72
A-3	Arrow Branches	74
A-4	Arrows Showing Mutual Control	74
K-1	Cone of Projection Tangent to Earth's Surface	169
K-2	The Developed Cone in the Plane	170
L-1	Data Format for "msns" and "spec" File Records	173
L-2	Data Format for "req" File Records	174
L-3	Data Format for "div" File Records	175
L-4	Data Format for "map" File Records	176
L-5	Data Format for "icao" File Records	177
N-1	Sample Plot of Map Coordinates and Selected Airfields	205
N-2	Sample of Printed Mission Data	206
N-3	Sample Plot of Mission Data	207
N-4	Sample of Printed Diversion Options Presentation	208
N-5	Sample of Plotted Diversion Option Presentation	209

LIST OF TABLES

Table		Page
I	Global Variables	39

ABSTRACT

The Military Airlift Command operates numerous C-5, C-141 and C-130 missions daily over worldwide air routes. Many of these flights operate partially full, and represent a considerable airlift capability if their scheduled routes and spare cargo space can be matched with unscheduled cargo or passenger requirements. Using existing data processing equipment at Headquarters Military Airlift Command, an automated system was designed and implemented to match these new airlift requirements with scheduled missions, and to compute and display the data necessary for rescheduling decisions. The system transfers scheduled-mission data from a Honeywell H6080 computer installation to a Hewlett-Packard 9825A Programmable Calculator and associated disk storage. System programs on the HP9825A select the most efficient missions for re-routing to accomplish an additional cargo or passenger requirement, graphically displays the change in routing on maps, and presents printed data on flying hours and monetary costs for each proposed mission diversion. Techniques derived from Structured Analysis and Design Technique and Structured Design methodologies were used to define and structure the system. A major objective of the system design and implementation was to assure the system users would be able to understand and modify any portion of the software to change or improve performance.

I. INTRODUCTION

This paper presents the design and implementation of a small management information system using a general-purpose programmable calculator, along with its peripheral equipment, to perform aircraft operations analysis and specialized geographic displays of aircraft operations data. The objective was to produce and geographically portray management data supporting decisions on short-notice changes to Military Airlift Command (MAC) cargo and passenger aircraft schedules.

The equipment available for use on the project included a Hewlett-Packard 9825A Desk-Top Calculator, 9872A Graphics Plotter, Impact Printer, and a 2640B Cathode Ray Tube (CRT) Terminal. In addition, a low-speed asynchronous data port on the World Wide Military Command and Control System (WWMCCS) was available as a source of data on MAC aircraft operations schedules.

A top-down development procedure used for this project. Selected portions of Structured Analysis Design Technique (SADT) conventions were major tools in organizing and isolating the major subtasks and functional activities.

The remaining sections of the Introduction provide background material on MAC's airlift scheduling, outline the objectives of the management analysis, system design and implementation, and present an overview of the remaining chapters of the thesis.

Military Airlift Command Opportune Rescheduling

In peacetime, Military Airlift Command strategic and tactical resources are used to move Department of Defense passenger and cargo requirements. Since the DOD's cargo and passenger requirements exceed

MAC's airlift capacity at peacetime activity levels, the efficiency with which MAC uses its peacetime flying hours for productive airlift determines the extent of residual DOD cargo and passengers which must be moved by commercial means at additional expense. With large numbers of flights daily, and with large sums of DOD funds at stake, there is a practical need for faster and better methods to analyze and alter MAC flight schedules to effectively respond to changing cargo and passenger requirements.

Prior to the design and implementation of the system presented in this thesis, experienced operations-management personnel at Headquarters MAC manually surveyed daily flight schedules searching for ways to alter an existing flight to accomplish an unforeseen or opportune airlift requirement. ("Opportune" is a term used informally within MAC to describe an airlift requirement which will be accommodated by more efficient scheduling of airlift resources already committed to other tasks). A reasonable alteration of an existing mission prevents adding a totally new mission to the schedule. Since most unforeseen or opportune airlift requirements allow only short planning lead-times, automated methods for collecting, analyzing and portraying data on mission diversion options are desirable.

An automated method to analyze scheduled mission data cannot accurately emulate the judgment possessed by operations-management personnel. Much of that judgment is based on factors learned through years of experience, and is not easily quantified or duplicated by automated means.

There are many, many factors which may be of critical importance in order to successfully select and reschedule a mission to pick up and deliver an additional cargo requirement. The operations-management personnel interviewed at Headquarters MAC were virtually unanimous in their belief that automated methods for rescheduling should not initially attempt to consider and apply all of these factors, but rather should seek to eliminate as many missions as possible from consideration through the use of those factors which may be predictably applied by the computer. An analysis of these factors and how they relate to management decisions on mission rescheduling is critical to any effort for improving the rescheduling process.

The Objectives of the Management Analysis

The objectives of the management analysis were to investigate and analyze the rescheduling tasks and the information needed for those tasks, and to look for ways to reduce the time required for the tasks, or to provide additional information for better rescheduling decisions. From the system viewpoint, the intent of the analysis was to reveal the appropriate point in the spectrum of alternatives between the existing rescheduling procedures at one extreme, and the ideal situation where an automated system accomplishes all of the rescheduling tasks with virtually no human intervention. In practical terms, this objective translates into finding practical methods to assist or enhance manual rescheduling tasks by automated applications on the available equipment.

The management analysis consisted of three distinct activities. First, an unstructured investigation was conducted to determine what needed to be done, and why, in order to effectively reschedule missions

against opportune cargo and passenger requirements. Second, these findings were organized into separate functional activities and information inputs and outputs. This, in other terms, became the requirements definition phase of the project, and is addressed in Chapter II. Third, the most practical means to accomplish the requirements had to be determined. This became the system design phase of the project as discussed in Chapter III.

The Objectives of the System Design

The results and documentation of the management analysis are provided by the SADT diagrams of Chapter II. The diagrams provide a simple, graphic representation of the functional activities involved and their relationships. Because the diagrams are simple and graphic, the users and system designer may use them as an effective means of communication regarding detailed system requirements. In addition, using the SADT conventions forces a complete and accurate perspective of system requirements, and has proven to be an effective basis for a system design structure (Ref 23:2-4).

The objectives of the system design process were to insure that the final system would meet the user's requirements, that it could and would be implemented within the time and technical constraints imposed by this thesis project, and to assure that the users would be able to maintain and modify the system as necessary. The last objective implies a well-structured and understandable system, with a functional, rather than technical, approach to the problems involved. For this reason, the SADT-derived hierarchical model of functional activities is particularly appropriate as a modular design structure for the system.

Overview of the Thesis

This paper presents the complete documentation for requirements definition, system design, and the hardware and software implementation. Chapters II and III present the requirements definition and system design. Chapter IV addresses the problem of transferring data from the WWMCCS system to the Hewlett-Packard calculator and storage disk. Chapter V explains the software implementation of the system design. The thesis concludes with a discussion of implementation results, and recommendations for possible improvements in the design and implementation process. Appendices to the thesis provide the detailed documentation of the system's software.

II. REQUIREMENTS DEFINITION

The requirements definition phase of system development is chiefly a form of dialogue between the system user and the designer as to what must be accomplished. To assure accuracy of communication, a language capable of more precision, descriptiveness and brevity than English prose has proven invaluable in the requirements definition for many large, complex systems. Several such "languages" have been developed, each with its advantages, appropriate applications, and drawbacks. After due consideration, one such language has been chosen for this project. The next section explains why this particular language was chosen. The last section of the chapter presents the requirements definition in the language chosen.

Structured Analysis as a Requirements Definition Language

A good requirements definition language must be both descriptive and precise so as to avoid ambiguities or omissions in the communication process. Further, a good language should be easily understood by all concerned in the requirements definition, since the eventual system users should be able to agree that the requirements definition document describes what they really want. In addition, the designers must be able to understand what is needed. Finally, an appropriate language for defining a particular system's requirements must lend itself to easy verification of completeness and accuracy. Completeness minimizes the threat of an unexpected problem late in the design or implementation process, while accuracy assures the design and implementation will accomplish system objectives.

Structured Analysis (SA) was chosen as the language for use on this project. It uses precise, well defined, graphic notations which are simple and quickly understood without an elaborate learning process. Since SA system models are built top-down, an omission of significant details are easily noted and the models therefore tend to be complete. The graphic conventions of the language force attention on areas lacking accuracy. However, the most important criteria for the choice of SA was based on the heuristic decision of applicability. While several other well-defined languages exist, they are not as simple and easily understood as SA, and they would be unnecessarily complex and unwieldy for a project of this size.

Structured Analysis conventions are described in several publications produced by Softech (Ref 23). Appendix A gives a short overview of the major points. For this project, the SA convention of developing both activity and data models for the system was omitted for brevity; only an activity model was developed. A project of this size does not need the additional checks for consistency provided by a data model.

Structured Analysis Activity Model

As an introduction to the Structured Analysis activity model, the objectives of Opportune Rescheduling for MAC cargo and passenger missions are reviewed. The objectives of the system are to automate, where practical, the tasks involved in reviewing MAC scheduled missions, selecting the most efficient mission to reschedule for an additional airlift requirement, and geographically displaying the pertinent mission information for management decision. Several practical constraints were mandatory considerations in the definition phase:

(1) Scheduled airlift mission data was available in digital form only on the Hq MAC World Wide Military Command and Control System (WWMCCS) implemented on a Honeywell 6080/6060 computer system.

(2) Operational time response required for use of the proposed system precluded use of the WWMCCS computer for this application. The time response requirement is not reflected in the SA diagrams.

(3) Graphic presentation and data processing capability for this system could only be effected by use of an available Hewlett-Packard 9825A calculator and its associated peripheral equipment.

The purpose of the SA activity model is to define the rescheduling system requirements in sufficient detail to assure understanding between user and designer, and to do so in such a manner that implementation of the requirements are technically feasible, and within the necessary constraints. An index of the model is provided in Fig 1, and can be used as an overview of the functions the system must perform. Two points concerning the model are noteworthy. First of all, Node diagrams labeled A-1 and A-2 are provided to put the system objectives in context with the environment in which it must operate, and to show sources and uses of the data involved in the system. Second, the activity model was not continued to the level of detail whereby each activity block became trivial, but rather to the level where both the system users and the designer were assured that the requirements were adequately defined.

System Purpose

To assist the reader in comprehending the requirements on a more global basis before reviewing the detailed documentation, the following summary of the system purpose and statement of required functions is provided.

A. System Purpose: A system of selecting and graphically displaying the best rescheduling options for new cargo and passenger requirements.

B. Required System Functions:

(1) Select, process, and store appropriate data on scheduled missions using WWMCCS computer facilities.

(2) Transfer stored data to Hewlett-Packard minicomputer for analysis, processing and graphics display.

(3) Determine most efficient options for rescheduling a mission to accomplish additional airlift, based on cost, flying hour program impact, and other considerations.

(4) Display options on geographically-oriented graphics display.

(5) Present pertinent management data on mission diversion options.

(6) Enable selective use of graphic and text display capabilities for related management information presentations.

A-2 Manage Airlift Operations

A-1 Plan and Direct Airlift Operations

A-0 Develop Mission Rescheduling Options and Costs

A1 Select Diversion Eligible Missions

A2 Pick Most Efficient Missions for Rescheduling

A3 Print/Plot Selected Data

Fig 2-1. Index to Structured Analysis Diagrams

A-2 Manage Airlift Operations (A-2)

In this context, management of airlift operations centers around the planning, scheduling, and directing actual aircraft operations in the most efficient manner possible. Airlift requirements, requests, and forecasts form the basis for planned airlift operations, and are constructed by A-21 into general schedules for various categories of airlift, and into management data needed by A-22 to allocate and schedule specific missions to the operational units (A-23). The operational units take allocated and scheduled missions, make the necessary plans and detailed arrangements to operate the missions, and thus complete the integrated airlift schedule. Since the schedule is initially based on forecasts and requests, the entire process is iterative in nature. Updated forecasts and changes in requests and requirements necessitate changes to the schedule. Since the schedule is reposited in a central computer (WWMCCS) data base, it is available at all levels in its latest form as changes to the requirements, and their corresponding missions, arise.

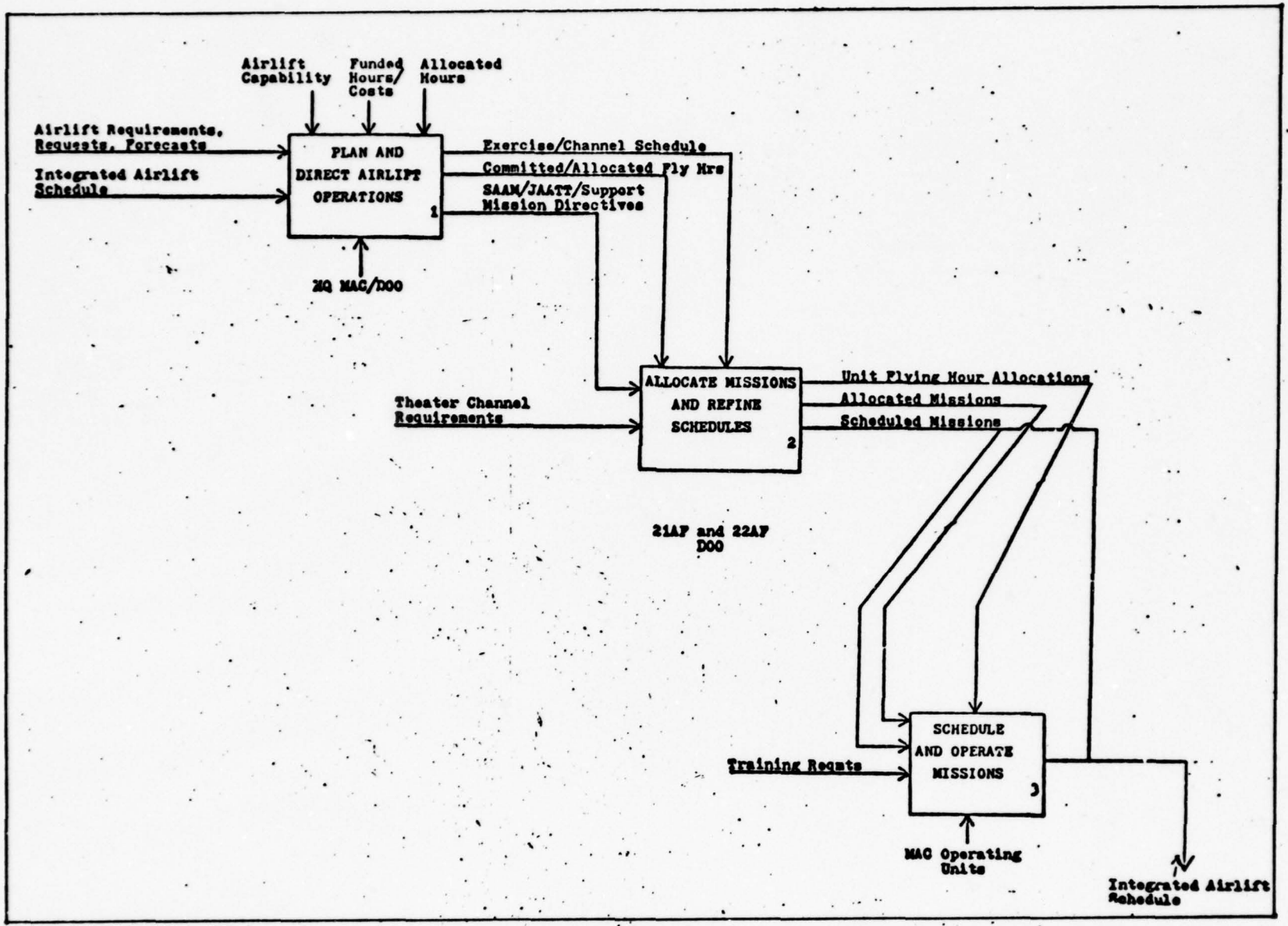


Figure 2-2. A-2 Manage Airlift Operations

Plan and Direct Airlift Operations (A-1)

Planning and directing airlift operations (A-1) is the most general and encompassing of the airlift operations management functions. Requirements and requests for exercise, Joint Airborne/Air Transportability Training (JA/ATT), and Special Assignment Airlift Missions (SAAM) are constructed into general schedules by A-11, A-12 and A-13 to be refined and detailed by lower organizational echelons. These general schedules are integrated with channel airlift forecasts, balanced with the capabilities of airlift resources, and formed by A-14 into the basic structure for the integrated airlift schedule. New airlift requirements are evaluated by A-14, and those which clearly require adding more airlift capability to the schedule are planned accordingly. Those which might possibly be accommodated by more efficient scheduling of existing missions are referred to A-15 for investigation and development of rescheduling options and their impact on fleet efficiency. This information is input to A-13 and A-14 for decisions and schedule updating.

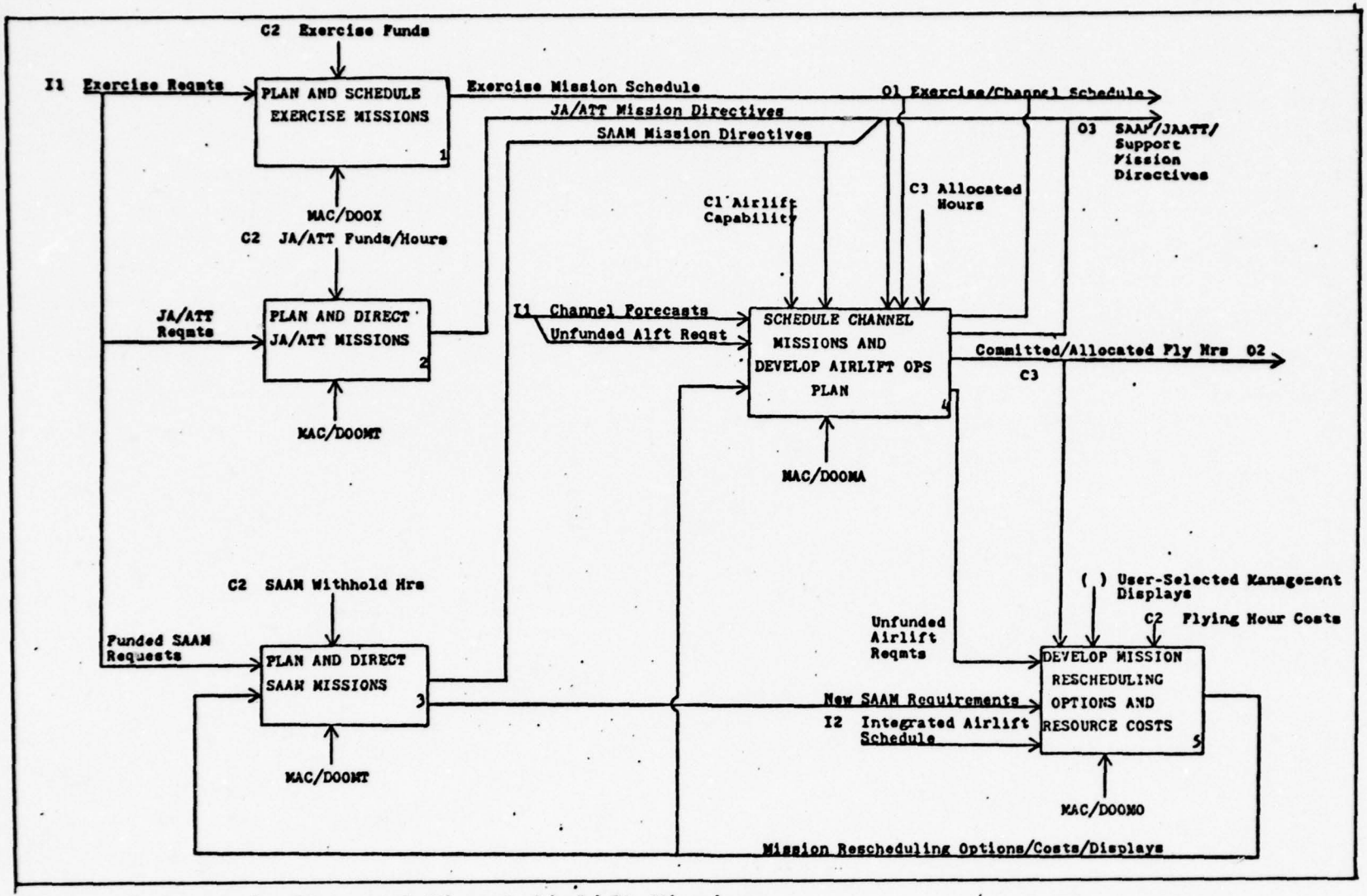


Figure 2-3. A-1 Plan and Direct Airlift Missions

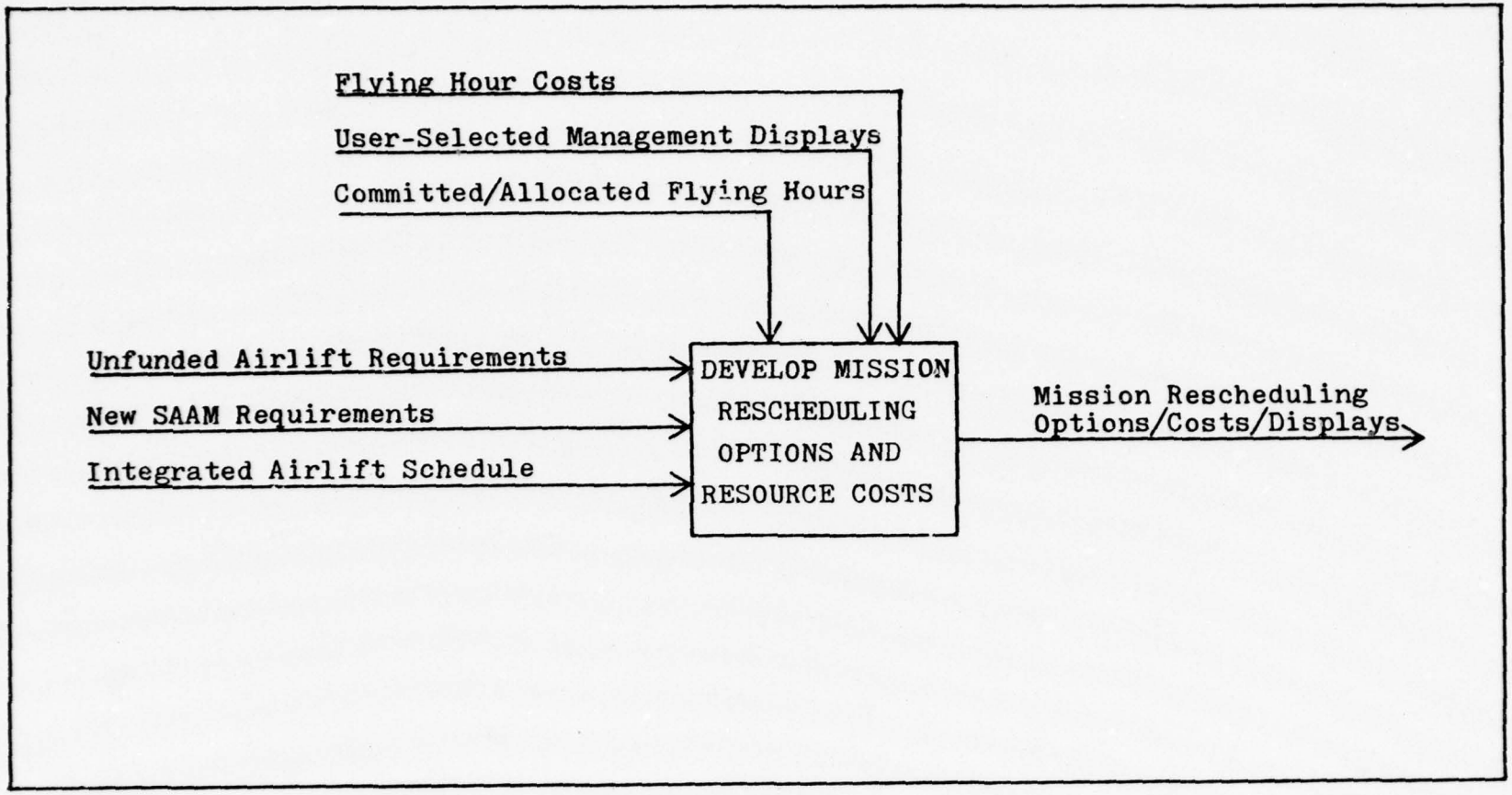


Figure 2-4. A-0 Develop Mission Rescheduling Options/Costs/Displays

Develop Mission Rescheduling Options and Resource Costs (A-Ø)

Integrated Airlift Schedule data maintained in the WWMCCS computer central data base must be analyzed by AØ1 to select scheduled mission legs which could legally and practically be rescheduled to accomplish additional cargo requirements. Data on these diversion-eligible missions must then be further analyzed by AØ2, in concert with a specific airlift requirement, to pick the mission or missions which could be most efficiently used to handle the requirement. When these missions have been picked, the information on the missions, the requirement, and the parameters of efficiency must be presented by AØ3 in a manner meaningful to the managers responsible for rescheduling decisions. In this case, the managers are chiefly personnel with aircrew backgrounds, and a graphic, geographical representation, in addition to printed data, is considered to be the most effective means of presenting the results.

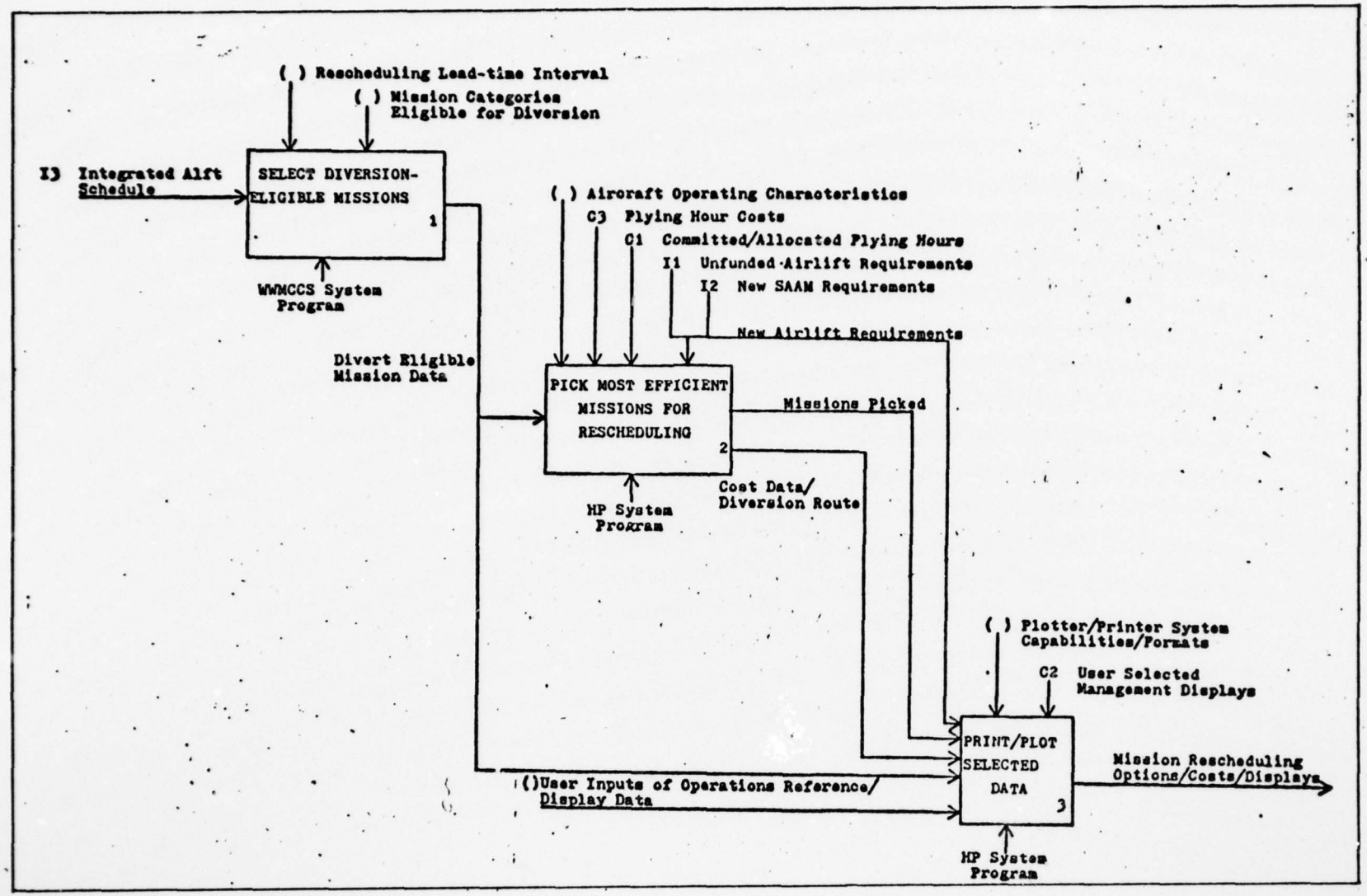


Figure 2-5. A0 Develop Mission Rescheduling Options and Resource Costs

Select Diversion Eligible Missions (A1)

Certain categories of individual mission flights may be identified in the data base as being eligible for diversion to any hypothetical airlift requirement. Since a mission may be comprised of several individual flights, or mission legs, a given flight may not be selected and diverted without considering the impact on those legs scheduled before and after the possible diversion. A11 selects mission legs which are eligible for diversion, A12 reconstructs the entire mission of which the mission leg is a part, and A13 and A14 compute parameters which may be used by A15 to determine if a particular mission leg could be diverted without adversely affecting accomplishment of the remaining portions of the mission.

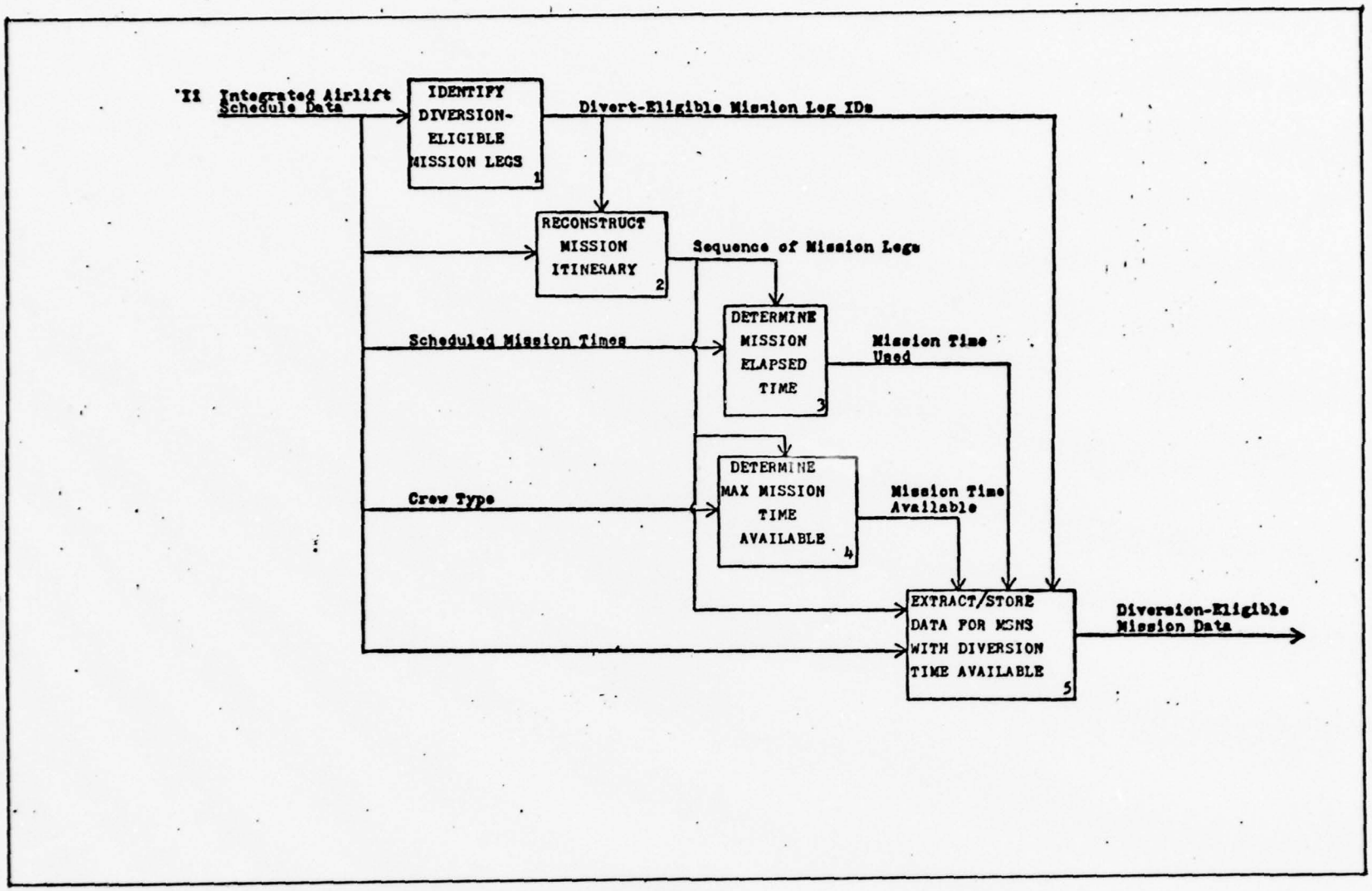


Figure 2-6. A1 Select Diversion Eligible Missions

Pick Most Efficient Missions for Rescheduling (A2)

Selecting the most efficient mission for a particular requirement involves several sequential steps. First of all, the mission and the requirement must be compared by A21 to insure compatibility among such factors as mission time and requirements delivery date, cargo space and cargo to be carried, type of aircraft and the airfield characteristics where the cargo must be picked up, etc. By then computing the distance and flying time involved in diverting the particular mission to accomplish the additional requirement, A22 may provide A23 the information necessary to decide whether the mission is capable of meeting its originally scheduled requirements if the added time is included. For those missions which can do so, A24 computes cost and selects parameters used by A25 to pick and store the most economical and practical missions for presentation to decision makers.

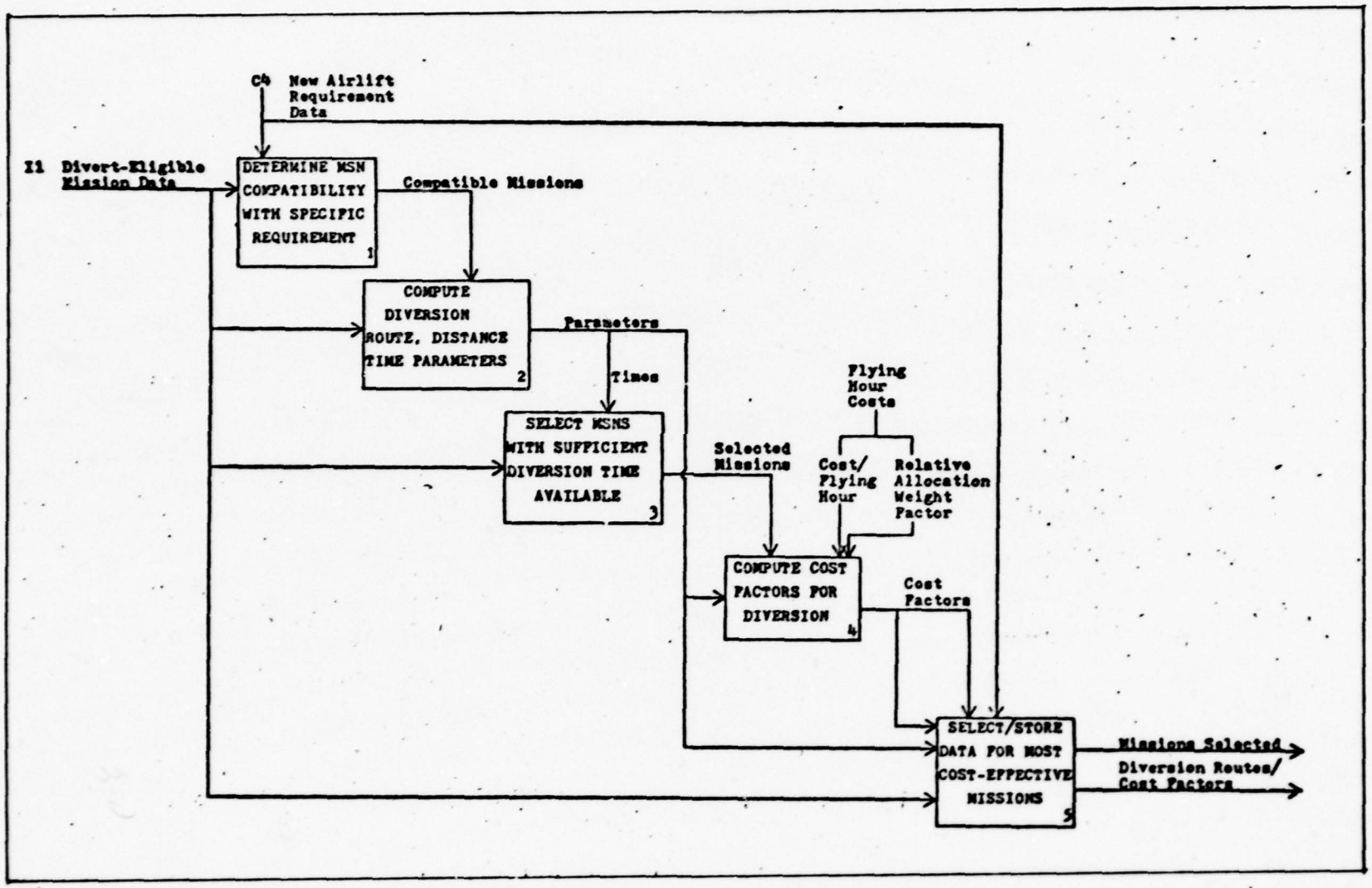


Figure 2-7. A2 Pick Most Efficient Missions for Rescheduling

Print/Plot Selected Data (A3)

Plotter and printer functions, and the data to be plotted or printed, are selected by the user according to specific system capabilities and the user's particular requirements. All plotter functions are to be geographically oriented. A31 interprets user commands and passes the necessary control data to effect the required response. A32 issues prompt messages to the user, and collects/formats/stores user data inputs which will partially or completely define the plotted or printed outputs. When a data source selection is made by the user, control of A33 is initiated to find the data identified by the user; A35 computes plotter coordinates, if plot is selected, and A36 plots the data according to the plot format for that particular type of data. A34 is initiated by A31 when plotting is selected, and computes the necessary parameters to be used by A35 for the particular user's choice of displays. A36 formats and issues printed output data according to the type of data selected. The output of A36 and A37 comprise the objectives of the system.

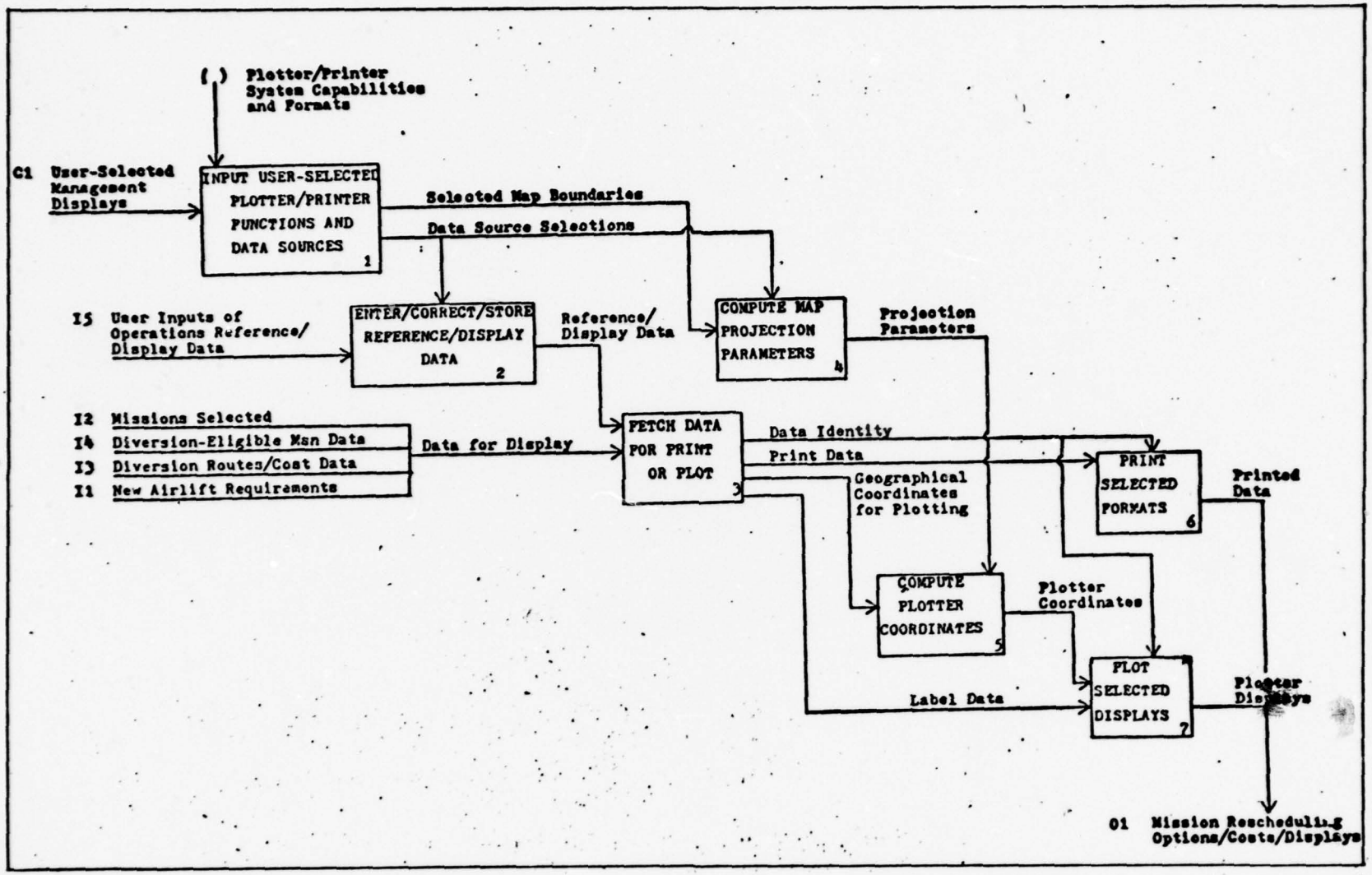


Figure 2-8. A3 Print/Plot Selection Data

III. SYSTEM DESIGN

In the design phase, the functions identified in the requirements definition are organized and structured according to certain principles known to enhance economy, reliability and maintainability of the system. Several design methodologies were considered for the system; some were rejected because they were, in their entirety, inappropriate and impractical to the somewhat limited size and complexity of this application. A composite of two design methodologies was chosen for this project as the most effective manner in which to apply proven software-engineering principles.

Primary Design Methodology

The primary methodology used in the design was an extension of Structured Analysis Design Technique. Since the requirements were defined using structural analysis (and many of the graphic conventions of SADT), the organization and functional hierarchy of the requirements definition could effectively be used to form the basic structure of the system. Conceptually, this is the SADT approach to software design. Because there were practical limitations imposed by the equipment available to implement the final design, some departures from the strictly functional hierarchy of the structured analysis organization could be anticipated. However, most of the functions represented by the SA activity charts could be implemented as distinct software modules, thus preserving the simplicity, accuracy and conceptual consistency achieved during the requirements definition. This procedure is not strictly within the SADT conventions,

but rather is the method chosen to exploit the functional activity and data relationships developed in the SA diagrams.

Additional Design Considerations

To determine when and how departures from the SA hierarchy should be planned, the principles of module coupling and cohesion from the Structured Design methodology were applied. Appendix 2 summarizes the important aspects of structured design, and briefly explains the principles of module coupling and cohesion. These principles are effective criteria upon which to evaluate and select the changes to the SADT-derived design necessary to overcome equipment limitations, and to better achieve the following system goals (Ref 5):

1. Understandability
2. Reliability
3. Maintainability
4. Modifiability
5. Generality
6. Usability
7. Efficiency

Both of the design techniques use the "top-down" approach to determine system structure. However, the limitations of the existing equipment dictated a departure from the top-down approach in several instances, for the exact manner in which some functions could be implemented on the HP calculator and its peripherals had a very definite impact on the final design. Limited calculator memory and disk storage, interface capabilities with the WWMCCS computer, and characteristics of the mechanical plotter, forced the construction of some modules from a bottom-up approach.

The modules which were so constructed, however, were designed to meet and interface with the top-down structure as low as possible in the overall hierarchy. The objective was to incorporate the best of the top-down and bottom-up approaches, and to strictly apply the principles of coupling and cohesion at the points in the module hierarchy where the two approaches interface.

The Basic Design

Fig 2 reflects the basic design in block diagrams. Block 1 is designed to be implemented as a program on the WWMCCS computer. Blocks 4, 5, 6 and 7 are designed to be implemented as overlays, controlled by Block 3, on the Hewlett-Packard calculator. Block 3 also performs the function of SA diagram Block A31, that of accepting and interpreting user command inputs. Block 2 partially corresponds to SA diagram A15, since this module controls reading the results of the WWMCCS processing and transferring/storing the data on the HP disk. However, its function is not distinctly reflected on the SA diagrams, since it exists to provide the practical requirement for a data link between the two computers. Block 4 corresponds to SA diagram block A2; Block 6 performs the function of SA diagram A32, and is designed to be implemented as a separate overlay due to the memory requirements necessary to accept and store user data inputs. Block 5 is functionally related to Block 7, since both provide data displays, and in the original design the two were grouped as one overlay to perform the functions of SA diagram blocks A33 thru A37. However, calculator limitations did not allow all the code necessary to implement these functions to reside in memory as a single overlay. Because the entire design, and particularly the interfaces between the top-down and bottom-up designed modules, would be impacted by a separation of the functions according to

the SADT activities, an heuristic approach to the division of this module was used. Since plotting of map geographical and political boundaries was the only activity independent and disassociated from the other displays in both its data source and probable sequence of user selection, this portion of the display function was constructed into a separate overlay. This division was feasible, and promised to minimize exchanging overlays during system operation.

Subsequent chapters explain the structure and modules used to implement each of the Hewlett-Packard calculator programs and overlays. The WWMCCS computer program to select diversion-eligible missions from the Airlift Integrated Management System (AIMS) schedule is not modularized due to the nature of the language selected. A high-level retrieval language, INQUEST, was selected for this program since it affords the user a capability to modify and maintain this part of the system without assistance from outside agencies. The code for this program is listed in Appendix D; it is a straightforward implementation of the requirements shown on SA diagram A-1. It could be as easily implemented in COBOL, FORTRAN or any of several other languages. The code for the Hewlett-Packard programs is listed in Appendices E through J. It is written in Hewlett-Packard Language (HPL), which is the only language acceptable to the HP 9825A calculator. HPL is a higher-order language semantically and functionally similar to FORTRAN; FORTRAN users should have little difficulty reading and understanding it after a review of the HPL notations provided in Appendix C. References 6 through 14 provide detailed specifications for HPL.

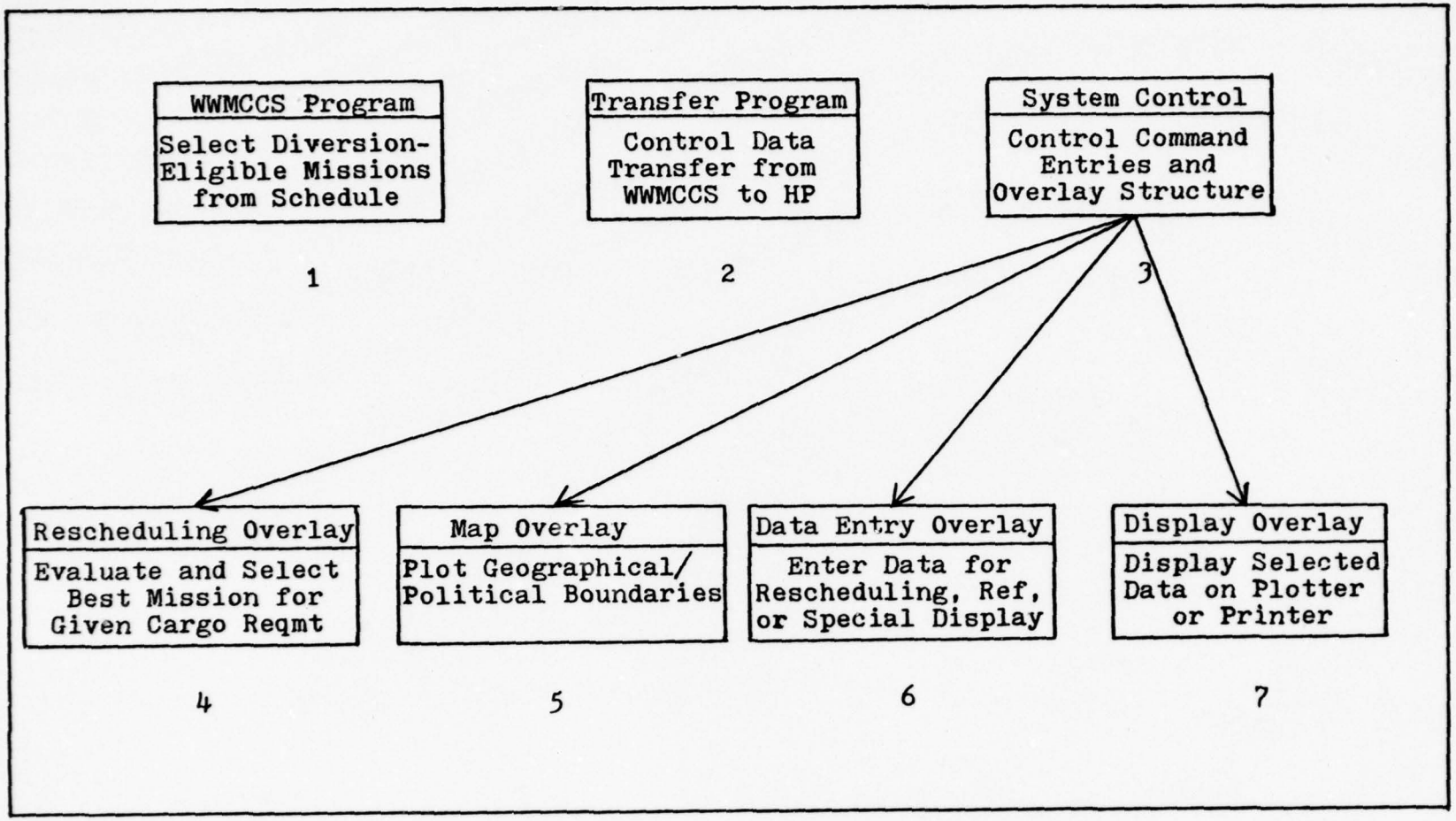


Figure 3-1. Basic System Structure

IV. THE DATA TRANSFER PROGRAM

The data transfer program is an integral part of provisions to input existing mission data into the Hewlett-Packard 9825A calculator. Since this is a physical function, and not a conceptual one, it has not been included as a part of the formal requirements definition, nor as a part of the overall system design except to allocate a "module" in the design to accomplish the function.

The data transfer function is necessary because of the physical constraints in which the system must operate. Military Airlift Command scheduled mission data, maintained by the WWMCCS data-base management system, must be processed, analyzed and displayed by the Hewlett-Packard 9825A calculator and its peripheral equipment. To effect such a requirement, only a limited number of alternatives are available. Manual entry of the mission data into the calculator from a WWMCCS system printout is totally impractical, since transfer of an estimated 50,000 characters of data on a daily basis is involved. It would be possible to have the WWMCCS system punch the data into cards, and then to read the card data into the calculator for storage on the Hewlett-Packard disk. However, the user's Hewlett-Packard equipment does not include a card reader, and the expenditure of additional funds for a card reader is undesirable unless no other alternatives exist. A serial data link between the WWMCCS system and the calculator is feasible, and is considered to be the most practical solution if a method of implementation can be effected.

Both synchronous and asynchronous WWMCCS system peripherals, over serial data lines, are available to the users (Ref 17). By using the

calculator to emulate one of these terminals, the WWMCCS mission data file may be accessed and read into the calculator, and then stored on the Hewlett-Packard disk for subsequent processing and display (Ref 13: 58). While the calculator cannot be used to emulate a synchronous terminal at the data transfer rates in use, emulating an asynchronous terminal at low or medium speed data transfer rates is within the calculator's capability (Ref 10:15-17).

Data Communications Conventions

The WWMCCS system terminals at Hq MAC are configured to operate on MIL-188 standard for serial data communications (Ref 17). The serial interface marketed by Hewlett-Packard for use with the 9825A calculator is capable of operation with only the Electronic Industries Association (EIA) RS-232-C standard for voltage and logic levels (Ref 13:12-13). Since the two standards are not compatible, a bi-directional converter is required to change from one standard to the other before a data link could be implemented.

A converter produced by Honeywell for military applications was procured and configured according to the vendors documentation for the required conversion.

Teletype Emulation

The WWMCCS system asynchronous terminal available to the users is an AT-33 teletype. An HPL program will allow the 9825A calculator to emulate the teletype, with the calculator keyboard used to send WWMCCS system time-sharing commands to access and read the mission data file.

A time-sharing command to list the data file to the terminal will get the mission data to the calculator in serial fashion. With the

Honeywell MIL-188 to RS-232 converter in the circuit, the data voltage and logic levels would be compatible with the Hewlett-Packard 98036A serial interface. Bytes of ASCII-coded mission data will be collected and reformatted by the HPL teletype-emulation and data-transfer program as necessary for storage on the Hewlett-Packard disk.

The Program

To effect the teletype emulation, data transfer and disk storage, the HPL program must accomplish the following:

a. Configure the 98036A serial interface for the proper parity, number of stop bits, and number of data bits per byte.

b. Establish a buffer in the calculator memory into which incoming bytes of data may be stored in the calculator pending processing.

c. Establish two levels of interrupts for incoming data: one interrupt routine must initiate reading a byte of data from the interface into the buffer when the byte is received; the other routine must properly identify and process the data in the buffer when an WWMCCS time-sharing message or data record is received.

d. The interrupt processing routines must display WWMCCS system messages on a printer or CRT, and must collect and reformat mission data and store it on the Hewlett-Packard disk.

e. Disk space for mission data must be initialized before each transfer to assure that the end portions of a lengthy mission data file from one transfer operation are not still stored on the disk after the subsequent transfer of a shorter file overwrites the disk space.

f. WWMCCS system commands entered from the calculator keyboard must be displayed at the calculator, and must be written to the serial interface for serial transmission to the Honeywell mainframe.

g. Hewlett-Packard keyboard-coded data must be converted to ASCII code before transmission to WWMCCS.

The transfer program accomplishes all of the above functions, although not in a modular fashion. The program code, and a brief explanation of which code segments are used to perform the functions, is listed in Appendix E.

Divert-eligible mission leg data records from the WWMCCS file are 61 characters long. The first character is an ampersand, and is used by the interrupt processing routine to differentiate between mission data and WWMCCS time-sharing messages. The remaining 60 characters are mission data in the following format:

Characters 2 - 4	Mission Sequence Number
Characters 5 - 7	Mission Subsequence Number
Characters 8 - 19	Mission ID
Characters 20 - 22	Type Aircraft
Characters 23 - 26	Departure ICAO
Characters 27 - 30	Departure Time
Characters 31 - 33	Departure Julian Day
Characters 34 - 37	Arrival ICAO
Characters 38 - 41	Arrival Time
Characters 42 - 44	Aircraft Configuration Code
Characters 45 - 47	Crew Type
Characters 48 - 49	Diversion-Eligible Code
Characters 50 - 60	Cargo and Passenger Data

When the interrupt routine identifies a sequence of characters received as mission data, it converts the data into parameters or strings,

then reformats the sequence of parameters and strings to match that of the mission file on the Hewlett-Packard disk. This data format is shown in Appendix L. When enough mission data has been received to fill a disk record, that record is written to the mission file.

The transfer program operates independently of the other HPL programs and overlays. Variable assignments in the transfer program are not consistent with the Opportune Rescheduling System program and overlays.

The next chapter explains the HPL implementation of the conceptual functions and activities identified in the requirements definition phase, and the structure of these functions as determined by the system design phase of the project.

V. THE OPPORTUNE RESCHEDULING SYSTEM PROGRAM

The functional requirement to select diversion-eligible missions from the AIMS scheduled-mission data base, and the physical requirement to transfer this data to the Hewlett-Packard storage disk, was explained in the previous chapter. This chapter addresses the remaining functions and activities identified in the requirements definition, and discusses the data storage and program modular structure used to implement these functions. The HPL program which implements these functions and activities is called the Opportune Rescheduling System.

To perform the conceptual functions and activities within the limitations imposed by the equipment, the Opportune Rescheduling System program must also perform certain program control functions as well. After accepting and interpreting user inputs to select specific system functions, a part of the program must control the loading of overlays and invoking the activity modules to accomplish the selected function. The manner and sequence in which the program's activity modules are invoked is determined by the hierarchy of functional activities identified in the requirements definition and system design phases of the project. The manner in which overlays are controlled is determined by the manner in which the hierarchical module structure is divided into segments which will fit into the limited calculator memory. While this requirement was addressed during the system design, a more detailed explanation of the relationship between the functional hierarchy of activity modules and the composition of the overlays is basic to an understanding of how the system is implemented in HPL code.

The calculator has 23K bytes of memory, which, in general terms translates into approximately 700 lines of HPL code. This approximation is very rough, since the number of variables allocated by the program, the potential depth of subroutine calls, and the exact HPL syntax all affect memory requirements for a given number of calculator operations (Ref 9:18). The total number of HPL code lines needed to implement the modular design totals 1100 lines. To implement a workable overlay structure, HPL code to accept user inputs selecting system options and code to control the overlays containing activity modules must always reside in calculator memory; the remaining code must be segmented into the overlays so that calculator memory is not exceeded. Further, the division of activity modules into feasible overlays must remain functionally consistent with the hierarchical modular structure.

The next section describes the segmentation of HPL code into an overlay structure. The HPL code which always resides in the calculator memory during system operation, called the Executive Control Program, will be described in the subsequent section, followed by four sections devoted to describing each of the overlays. The remaining sections in the chapter will show the Opportune Rescheduling System detailed module hierarchy, and explain the conventions used for data file organization and program and data storage.

Overlay Structure

The structure imposed by the system design implies a segmentation of the HPL-coded activity modules into three functional divisions. Two of these are explicitly identified in the requirements definition; one is to pick the most efficient mission for rescheduling, and the other is

to print and plot the selected data (see Fig 2-5). For brevity, these functions and their corresponding code segments are termed "OPTIMIZATION" and "DISPLAY", respectively. The third function is not on a par with the other two in terms of computational complexity, but in terms of the number of lines of code and calculator memory requirements, is very nearly the equal of the other two. Entering, correcting, and storing reference or display data is hierarchically subject to the "DISPLAY" function (see Fig 2-8), but implementation of this function in HPL requires considerable calculator memory which, in turn, necessitates the distinct segmentation. For brevity, this segment is termed "ENTER".

Implementation of the "DISPLAY" function, in its entirety, requires more than the available calculator memory, and further segmentation of this function is necessary because of this constraint. The segment of code responsible for plotting map boundaries, grids and legends is the only portion of the code completely independent of other portions, so sub-division of the "DISPLAY" segment is based solely on the necessity for segment independence. This additional segment is termed "MAP PLOT".

The resulting overlay structure is one which is functionally consistent with the SADT design, yet also allows the program to perform a specific function without all of the HPL code residing in the calculator memory. Subsequent references to these four overlays will be in capital letters, in order to differentiate between functional modules, or subroutines, with the same name. An overlay contains all of the subordinate modules needed to implement a specific function; the module with the same name as the overlay is the subroutine in the executive control program that invokes the overlay modules. The next section describes the executive control program.

THE EXECUTIVE CONTROL PROGRAM

The program first performs the data management activities common to most programs. Variables are assigned, arrays dimensioned, and data disk files are identified. The user is then prompted to enter a function, ENTER, OPTIMIZE, or DISPLAY. The user's response invokes either the "enter", "opt" or "display" module. These modules prompt the user to select a data source, missions, requirements, ICAO identifiers (airfields), map coordinates, or special missions, and then loads the proper overlay and invokes a module within the overlay to accomplish the selected function.

There are also a number of procedural modules that perform detailed activities common to two or more of the basic functions, which are loaded with the executive control program. They are not immediately subordinate to the "enter", "opt", or "display" modules, but are generally much further down in the structural hierarchy. Loading these common-use modules with the executive control program, which always resides in memory, prevents duplicate code for these activities in the overlays, and facilitates module code changes and corrections.

The HPL code for the executive control program, and for the procedural modules loaded with the program, is listed in Appendix F. A description of the function of each module is also included in the Appendix. The remaining HPL-code modules are contained in the four overlays. The next four sections describe the overlay composition.

THE MAP PLOT OVERLAY

The Map Plotting Overlay consists of a hierarchy of modules which compute map projection parameters, establish map plotting conventions,

and produce stored map coordinates in a printed or plotted display format.

Map projection parameters, based upon simple conic projection concepts, are computed based upon the map boundaries selected and the size and aspect ratio of the plotting surface. Appendix K describes in detail the computation of map projection parameters; Table I describes the allocation of these parameters to calculator global variables for use by the rescheduling system program modules.

Map plotting conventions, such as the geographical boundaries for the map, size and aspect ratio of the plotting surface, and the pen colors to be used for the various data plots, are established by user inputs (or system default values). Table I also describes the allocation of these conventions to calculator global variables for use by the rescheduling system program modules.

The production of plotted or printed displays from stored map coordinates is accomplished by a hierarchy of functional modules invoked by the executive control program. The modules issue prompt messages and accept user inputs to identify which coordinates are to be displayed and whether the coordinates are to be plotted or printed. The selected coordinates are then printed, or mathematically projected and converted into plotter units and plotted by the HP 9872A Graphics Plotter.

The code for the map plotting is listed in Appendix G. A description of each module's function is also included in the Appendix. Map coordinates, as well as all other stored data used by the OPPORTUNE RESCHEDULING SYSTEM (with the exception of the mission data transferred

TABLE I

GLOBAL VARIABLES

r0 Output Device Address
r1 Current Pen Position (Longitude)
r2 Current Pen Position (Latitude)
r3 Current Pen Position (NM from Map Origin, X Direction)
r4 Current Pen Position (NM from Map Origin, Y Direction)
r5 Last Pen Position (NM from Map Origin, X Direction)
r6 Last Pen Position (NM from Map Origin, Y Direction)
r7 Spare
r8 Spare
r9 Map East Boundry (Degrees Longitude)
r10 Map West Boundry (Degrees Longitude)
r11 Map Mid Meridian (Degrees Longitude)
r12 Map North-South Midpoint (Degrees Latitude)
r13 Radius of Tangent Cone
r14 NM from Map Origin to Boundry in X Direction
r15 NM from Map Origin to Boundry in Y Direction
r16 Map Constant of Projection
r17 Plotter Left X-Limit (Plotter Units)
r18 Plotter Lower Y-Limit (Plotter Units)
r19 Plotter Right X-Limit (Plotter Units)
r20 Plotter Upper Y-Limit (Plotter Units)

TABLE I (cont'd)

GLOBAL VARIABLES

r21 Spare

r22 Map Grid Pen Color

r23 Map Pen Color

r24 Map Interrupt Flag

r25 Mission Plot Pen Color

r26 Requirement Plot Pen Color

r27 ICAO Plot Pen Color

r28 Diversion Plot Pen Color

r29 Mission Legend Flag

r30 Line Color Legend Flag

r31→r38 Interrupt Routine Variables

r39 Minimum Diversion Cost Factor

r40 Line Number Counter for Overlays

F[1,1] C130 Cost/Flying Hour

F[1,2] C130 Extra Flying Time/Stop

F[1,3] C130 Evaluation Weight Factor

F[1,4] C130 Block Speed

F[2,1]→F[2,4] Data as above for C141

F[3,1]→F[3,4] Data as above for C5

from WWMCCS) is entered and maintained through capabilities provided by the ENTER overlay modules. The next section describes the ENTER overlay.

The ENTER Overlay

The ENTER overlay consists of a hierarchy of modules which provide the capability to add, change or delete mission, requirement, airfield, map coordinates and special mission data stored on the Hewlett-Packard disk. When the user selects this mode of operation, the system issues prompt messages which determine which data file is to be modified, and whether data is to be added, changed, or deleted. A specific ENTER overlay module corresponding to each data file and type of file modification is invoked to provide user prompt messages, accept data identification and new or changed data inputs, and perform the desired modification.

To achieve any degree of system reliability, accurate and well-structured data files are essential. Since each data file is organized differently, the modules which modify the data files were designed to hide as many of the file structure details as possible from the rest of the program (Ref 3). This approach strictly limited the number of modules which could potentially cause data file problems, and simplified the top-down programming, testing and module interfaces. The resulting structure is understandable and balanced.

Using the top-down design approach, two functions were identified as critical to data file accuracy. One was to get and accurately format data according to the file's data record structure. The other was to insure that data is stored in the proper location within the file. For

each type of data, a separate module was designed to accomplish each of these functions. The only information passed to the modules which get and store the data is the location within the file; prompt messages, user data inputs, edits, reformatting and writing data to the disk are all handled within the module.

The only information passed to the modules which find the proper file location is the data identity, or key. Since locating specific data within a file is also a necessary function for data retrieval, these modules are used for many system operations, and are included as procedural modules loaded with the executive control program.

The HPL code and a description of all the modules which comprise the ENTER overlay are located in Appendix H. The hierarchical structure of the modules is outlined in a later section of this chapter.

The DISPLAY Overlay

The DISPLAY overlay consists of a hierarchy of modules which, at the user's selection, retrieves stored data from the data files and displays the data in a plotted or printed text format. Data plots are all geographically oriented; the map coordinate and airfield data include geographical coordinates which may be mathematically projected and plotted as a map. Missions, airlift requirements, and diversion option data all include airfield references, and each concerns the movement of aircraft from one geographical location to another. Therefore, all stored data possesses a geographical relationship which may be meaningfully displayed in map form.

Each print module, by using subordinate modules, finds the data identified by the user, transfers it into calculator memory, reformats

it into ASCII bytes, and sends it to the CRT or printer. Each plot module, using a similar structure of subordinate modules, finds the data identified by the user, transfers it into the calculator memory, traces airfield references to stored coordinates if necessary, mathematically projects the coordinates to plotter units, and sends the appropriate commands to the HP 9872A Graphics Plotter to produce the display requested.

The mathematical projection of coordinates is based upon a simple conic projection (Ref 18:5). Appendix K describes the map projection mathematics in detail. The conversion into plotter units is based upon the map boundaries selected by the user. When the boundaries have been selected, the system scales the physical plotting surface into nautical mile units so that plotter command parameters are equivalent to nautical miles over the earth's surface. This allows use of the same parameters to compute distances for mission routes between two airfields. With the exception of map coordinates, plotted data is also labeled by the program. Again, the modules which perform the labeling hide all the implementation details involved; only the data file location is passed to these modules.

The HPL code for the DISPLAY overlay and a description of each module's function is listed in Appendix I. The hierarchical structure of the modules is outlined in a later section of this chapter.

The OPTIMIZE Overlay

The OPTIMIZE overlay consists primarily of a collection of serially-invoked modules which analyze different aspects of a diversion-eligible mission relative to a given airlift requirement, and selects and stores those which are the most economical diversion options. When the user

enters an airlift requirement, or selects one which has already been stored, the OPTIMIZE modules begin processing the mission data file by reading sequential disk records into calculator memory, and invoking activity modules which analyze a particular mission in a step-by-step process to determine if it may be feasibly diverted to the requirement.

If the mission can be diverted, more activity modules compute routes, distances, flying times and costs incurred. This data is stored as one of six diversion options if the computed costs are lower than any one of the previously-stored options. The system stores data on six options in order to present airlift managers a range of viable alternatives; a number of qualitative factors, not available as stored mission schedule data, but which may be equal in importance to economy will frequently enter into the final decision to divert a mission. For the same reason, the system has the capability to reanalyze the mission file and select the next six-best options.

The activity modules and the sequence in which they are invoked to analyze a mission were designed according to the functional definition of requirements depicted by SADT diagram A2 (Fig 2-7). Since the requirements definition effort considered all of the mission data which the Airlift Integrated Management System (AIMS) on WWMCCS was planned to provide, the design provides the modular structure to make use of all the needed data. AIMS, however, is still in the development stage, and reliable data on scheduled cargo, passengers, and aircrews is not yet available. The modules designed to use the unavailable data are, in the current configuration, dummy modules. They are coded so as not to preclude a mission from being selected as a diversion option, and it is

left up to the airlift managers to evaluate the diversion options selected and assure that actual cargo, passenger, and aircrew information is considered in the final decision. Since the modular structure and the functional and interface requirements for the modules are already provided, coding and implementing operational modules can be accomplished by the user when the AIMS data elements become available.

The code for the OPTIMIZE overlay modules, and a brief functional description for each module, is provided in Appendix J. The hierarchical module structure is outlined in the next section of this chapter. The last two sections of the chapter describe the program and data storage conventions for the OPPORTUNE RESCHEDULING SYSTEM.

SYSTEM HIERARCHICAL MODULAR STRUCTURE

The following outline depicts the hierarchy of the program and overlay modules. The letter in parenthesis after each module name indicates in which program or overlay the module is stored. (Ø = Executive Control Program, 1 = MAP PLOT Overlay, 2 = ENTER Overlay, 3 = DISPLAY Overlay, 4 = OPTIMIZE overlay.)

EXECUTIVE CONTROL PROGRAM

- mapsetup (1)
- mapintplt (1)
  - mapsetup (1)
    - pens (1)
    - grid (1)
    - border (1)
  - crecpltint (1)
    - proj (Ø)

```
        center (1)
           equ (1)
        crcplt (1)
           ang deg (1)
enter (Ø)
   cenmsn (2)
      addmsn (2)
         msnget (2)
            mcrt (Ø)
      delmsn (2)
         mfind (Ø)
         mert (Ø)
      chgmsn (2)
         mfind (Ø)
         mcrt (Ø)
         msnget (2)
            mcrt (Ø)
   cenreg (2)
      addreq (2)
         reqget(2)
            rcrt (Ø)
      chgreq (2)
         reqget (2)
            rcrt (Ø)
      delreq (2)
         rfind (Ø)
         rcrt (Ø)
```

```
cenica (2)
   addica (2)
      iget (2)
   chgica (2)
      ifind (Ø)
      icrt (Ø)
      iget (2)
   delica (2)
      ifind (Ø)
      icrt (Ø)
cencor (2)
   addcor (2)
      cget (2)
   delicor (2)
      ccrt (2)
   chgcor (2)
      ccrt (2)
      cget (2)
censpc (2)
   cenmsn (2)
      addmsn (2)
         msnget (2)
            mcrt (Ø)
   delmsn (2)
      mfind (Ø)
      mcrt (Ø)
```

```
chgmsn (2)
   mfind (Ø)
   mcrt (Ø)
   msnget (2)
      mcrt (Ø)
opt (Ø)
   weight (4)
      weightprt (4)
   rfind (Ø)
   rcrt (Ø)
   reqget (4)
   divset (4)
      rfind (Ø)
   optreq (4)
      rfind (Ø)
      mincost (4)
         cost (4)
            rfind (Ø)
         divset (4)
            rfind (Ø)
      eligtype (4)
         ifind (Ø)
      eligcat (4)
      eligtime (4)
      eligicao (4)
      eligcargo (4)
```

```
reqtime (0)
   ifind (0)
   proj (Ø)
mtime (Ø)
   ifind (Ø)
   proj (Ø)
onloadtime (4)
   ifind (Ø)
   proj (Ø)
offloadtime (4)
   ifind (Ø)
   proj (Ø)
divmaxtime (4)
   rfind (Ø)
divmin (4)
   rfind (Ø)
   cost (4)
      rfind (Ø)
optcrt (Ø)
   rcrt (Ø)
   crtline (Ø)
   mfind (Ø)
   mcheck (Ø)
   dcrt (Ø)
```

```
optdis (3)
   rfind (Ø)
   rcrt (Ø)
   optcrt (Ø)
      rcrt  (Ø)
      crtline (Ø)
      mfind (Ø)
      mcheck (Ø)
      dcrt (Ø)
   optlegend (3)
   optplt (3)
      rfind (Ø)
      rplt (3)
         ifind (Ø)
         iplt (3)
            proj (Ø)
            ilabel (3)
      dplt (3)
         'ifind (Ø)
         iplt (3)
            proj (Ø)
            ilabel (3)
         dlabel (3)
            mfind (3)
            mlabel (3)
               angdis (3)
```

```
            mfind (Ø)
            mcheck (Ø)
            mpltall (3)
               mplt(3)
                  ifind (Ø)
                  iplt (3)
                     proj (Ø)
                     ilabel (3)
                  mlabel (3)
                     angdis (3)
         optprt (3)
            optcrt (Ø)
               rcrt (Ø)
               crtline (Ø)
               mfind (Ø)
               mcheck (Ø)
               dcrt (Ø)
   display (Ø)
      mapsetup (1)
         pens (1)
         grid (1)
         border (1)
      pens (1)
      rdis (3)
         rfind (Ø)
         rcrt (Ø)
```

```
   rplt (3)
      ifind (Ø)
      iplt (3)
         proj (Ø)
         ilabel (3)
      rlabel (3)
         angdis (3)
   rprt (3)
      rcrt (Ø)
idis (3)
   ifind (Ø)
   icrt (Ø)
   iplt (3)
      proj (Ø)
      ilabel (3)
   iprt (3)
      ifind (Ø)
      icrt (Ø)
mdis (3)
   mlegend (3)
   mfind (Ø)
   mcrt (Ø)
   mplt (3)
      ifind (Ø)
      iplt (3)
         proj (Ø)
         ilabel (3)
```

```
      mlabel (3)
         angdis (3)
   mprt (3)
      mfind (Ø)
      mcrt (Ø)
mdisall (3)
   mlegend (3)
   mfind (Ø)
   mcheck (Ø)
   mcrt (Ø)
   mpltall (3)
      mplt (3)
         ifind (Ø)
         iplt (3)
            proj (Ø)
            ilabel (3)
         mlabel (3)
            angdis (3)
   mprtall (3)
      mcrt (Ø)
cdis (1)
   mapplt (1)
      proj (Ø)
      center (1)
         equ (1)
      arcplt (1)
         angdeg (1)
```

```
optdis (3)
   rfind (3)
   rcrt (Ø)
   optcrt (Ø)
      rcrt (Ø)
      crtline (Ø)
      mfind (Ø)
      mcheck (Ø)
      dcrt (Ø)
   optlegend (3)
   optplt (3)
      rfind (Ø)
      rplt (3)
         ifind (Ø)
         iplt (3)
            proj (Ø)
            ilabel (3)
      dplt (3)
         ifind (Ø)
         iplt (3)
            proj (Ø)
            ilabel (3)
         dlabel (3)
            mfind (3)
            mlabel (3)
               angdis (3)
```

```
            mfind (Ø)
            mcheck (Ø)
            mpltall (3)
               mplt (3)
                  ifind (Ø)
                  iplt (3)
                     proj (Ø)
                     ilabel (3)
                  mlabel (3)
                     angdis (3)
         optprt (3)
            optcrt (Ø)
               rcrt (Ø)
               crtline (Ø)
               mfind (Ø)
               mcheck (Ø)
               dcrt (Ø)
```

PROGRAM AND DATA STORAGE

The executive control program is stored on the Hewlett-Packard OPPORTUNE RESCHEDULING disk under the file name "contrl". It is also duplicated on the OPPORTUNE RESCHEDULING cassette tape cartridge on file Ø. Either may be loaded into the calculator for execution by using the appropriate HPL program fetch commands for disk or tape; overlay loading is automatically handled by the executive control program from the OPPORTUNE RESCHEDULING disk. The MAP PLOT overlay is stored on the disk under the

file name "onefil", the ENTER overlay in "twofil", the DISPLAY overlay in "arefil", and the OPTIMIZE overlay in "forfil".

HP disk data files are organized by named files, and by 256-byte records within each file. File size is established when the file is initialized by specifying the number of records needed. Data files for the opportune rescheduling system were sized to meet projected user requirements; they may be expanded at a later time if necessary.

All system data is stored on the OPPORTUNE RESCHEDULING disk also. Mission data transferred to the disk from the WWMCCS system is stored under the file name "msns"; special mission data under the file name "spec"; airfield data under the file name "icao"; map coordinates under the file name "map"; airlift requirement data under the file name "req"; and mission diversion data under the file name "div".

The manner in which each of these files is logically organized was based on the need for a simple, functional method to maintain data relationships. A more rigorous approach to this organization, using data base management techniques, should reduce the disk transfer and processing times significantly. Recommendations to this effect are included in Chapter X.

DATA FILE ORGANIZATION

The "msns" file consists of 200 disk records; each record stores data on six individual mission legs. A Military Airlift Command mission may consist of one or mission legs. Each mission is assigned a unique mission sequence number; every mission leg of that mission has the same sequence number, but is assigned a unique sub-sequence number. Mission legs for the same mission are stored together in the order in which they will be flown, whereas the overall file structure is organized under the

"pile" concept, which implies no specific order of storage (Ref 22:75-77). Retrieval keys for individual mission legs are the sequence number and sub-sequence number.

The "spec" file consists of only one record, allowing storage for up to six mission legs. Data storage format and file organization for the spec file is the same as for the "msns" file.

The "req" file consists of one disk record; data for up to nine distinct requirements may be stored. The retrieval key for requirement data is the requirement number, a value from one to nine. The file organization within the disk record is sequential by requirement number.

The "icao" file consists of 50 disk records; each record stores data on up to 25 individual airfields. The retrieval key for locating specific airfield data is the four-letter ICAO identifier. The file is organized under the pile concept, except that the initial storage of data in the file placed frequently-used airfields near the beginning of the file to speed up data retrievals.

The "map" file consists of 50 disk records; each record stores data on up to 42 individual map-coordinate plotting points. The retrieval key for map-coordinate data is the record number and an index number which identifies which of the 42 record segments is desired. The file is organized sequentially by the sequence in which individual data points must be plotted. Map coordinate data is stored as geographical coordinates for individual points, plus a code which indicates what kind of line the plotter must draw as it moves to that point.

The plotter moves from point to point in the same sequence that the coordinates are stored. A code of 1 directs the plotter to move to the

longitude and latitude specified without drawing a line; a code of 2 directs the plotter to draw a straight line from its previous position to the longitude and latitude specified. A code number greater than two directs the plotter to draw a counter-clockwise arc, with a radius in nautical miles equal to the code value, from the plotter's previous position to the coordinates specified. A negative code number invokes the same response, except the arc is in the clockwise direction. A code value of Ø is ignored by the plotter; it disregards any coordinates stored with the code, and remains at its previous position.

The "div" file consists of nine disk records; each record stores data on six diversion options for a given requirement. Each record number corresponds to the specific airlift requirement stored in the "req" file under the same number. The retrieval key for specific diversion data is the record number and an index to the specific record segment. The file is organized sequentially by the corresponding requirement number and the ranking of diversion options for that requirement.

The data storage format for each of the files is shown in Appendix L. The special mission file is provided to allow the user to enter and store unique mission data from the calculator keyboard, and to have it printed or plotted for special management display purposes. The data format is the same as for the "msn" file.

Data read into the calculator is stored in memory as string variables. An entire disk record is transferred into a 252-byte string variable for each read operation. Four bytes on the disk record are storage overhead and are not usable for data. Mission data or special mission data are

stored in the string variable M$; requirement in R$; airfield data in I$; map coordinates in C$; and diversion data in D$.

A number of parameters concerning user option selections and system performance are maintained in calculator memory as global variables for use by any of the activity modules. Table I identifies and defines these variables.

This section concludes the description of the actual implementation of the OPPORTUNE RESCHEDULING SYSTEM design. Implementation results and recommendations for system improvements will be discussed in the last chapter.

VI. IMPLEMENTATION RESULTS AND RECOMMENDATIONS

The primary objective of this thesis was to implement and document a functional system which would reduce the time and effort required to select and display mission diversion options. The secondary objective was to produce a system which was understandable, and one which could be maintained and modified by the users without ADP assistance. The preceding chapters have described the processes which determined what tasks or functions had to be realized to accomplish these objectives, how the necessary functions should be organized and structured, and specifically what was done to implement the functions. The process which governed how the individual functions were implemented and integrated into the overall system will be addressed in the next section.

Coding and Testing

The approach to coding and testing the system paralleled the design approach; that is, a combination of the top-down and bottom-up methodologies was used. The modules concerned only with what activities must be accomplished were designed with the top-down approach from the SADT diagrams. The modules concerned with how a function must be accomplished using the available equipment were designed from a bottom-up approach. Generally, the two groups of modules were coded and tested individually in the same order. The modules designed from the top-down approach were coded and tested from the top of the hierarchy toward their interface with the bottom-up modules. Dummy modules, or stubs, were used to simulate subordinate modules during the top-down testing. The bottom-up designed

modules interfaced with the calculator peripheral equipment; short test programs were used to assure that the equipment would respond properly when the modules were invoked.

To integrate the system modules, a combination of the "incremental" and "phased" testing and implementation strategies was used. Phased testing, or simultaneously testing several modules as a group, was used to test the modules within each overlay. The incremental approach, whereby the interface between every module is tested individually, and then in incrementally-larger groups, was used to test the overlays and their interaction with the executive control program. This combination of approaches simplified code debugging, since functional deficiencies within an overlay were easily traceable to a specific module, and it simplified system implementation since code testing and system integration were essentially one process.

System integration and testing results provided a reasonable basis to conclude that the primary objective of the thesis was accomplished. The actual system performance conforms to the designed objectives in every respect. Diversion-eligible mission data is selected from the AIMS data base, and successfully transferred to the Hewlett-Packard system. According to the specific criteria established by the users, the best mission options to divert to a given airlift requirement are chosen from the mission data. Data on missions, airlift requirements and diversion options may be selectively displayed as map plots or printed text.

An evaluation of the secondary objective, how well the system may be maintained and modified by the users, cannot be made on the basis of the

integration and testing. Structural simplicity, module functionality and documentation accuracy are system characteristics which should contribute to this objective; however, their effectiveness can only be measured over time by the user.

One aspect of system performance which surfaced during the testing, and one not specifically addressed by the requirements definition, degrades the effectiveness of the system from the users' standpoint. The time required to transfer data, and the time required to compute and select optimum diversion options, detract to some extent from the system's practicality. The data transfer requires an average of ninety minutes to complete; the selection of diversion options frequently requires a similar amount of time. Although the system operates without user interaction during those intervals, and still reduces considerably the human effort involved in rescheduling, there is an undesirable limit to the number of requirements which can be processed daily. Recommendations regarding possible methods to decrease the transfer and processing times will be addressed in the next section.

Recommendations for System Improvements

Based upon subjective judgments of the system made during the definition, design, implementation and testing phases of the project, recommendations for system improvements fall into two categories: reducing data transfer and processing times, and periodic enhancements to the diversion option selection process.

The time required for data transfer is directly attributed to the low speed baud rate (110 baud) of the serial data-communications link

between the WWMCCS and Hewlett-Packard systems. The WWMCCS system configuration for Military Airlift Command has the capability for asynchronous data communications at 300 baud. Installation of a dedicated line and modem capable of the higher rate would reduce the data-transfer time proportionally. The only modification required for the system to accommodate the higher rate is a change to the baud-rate selection for the HP Serial I/O interface.

The time required for processing the mission file is directly related to the length of the file, and to the amount of processing required to sequentially evaluate each mission. The processing time could be reduced significantly by a better file structure for the mission data file. Every mission in the file is evaluated for its eligibility in regard to the airlift requirement; if a mission fails any of the eligibility criteria, it is eliminated from further consideration as a diversion option. The file could be structured, however, according to these eligibility criteria, and only the eligible portion of the file would need to be read from the disk and processed.

There are many methods which could be used to structure the file and achieve the decreased processing time. A simple and easily-implemented method would be to sequence the file by mission arrival day and time, and then terminating the processing when the mission arrival times would preclude meeting the requirement delivery time. This method could be used for any of the several eligibility criteria.

A more complex, but perhaps more effective method to reduce processing time would be to compute and store tables which indicate which missions

fall within certain eligibility categories for all the eligibility criteria. For a given airlift requirement, a search of the tables would reveal which missions may be eligible and should be read from the disk for further evaluation.

Any such method to decrease processing time must be evaluated with respect to the memory requirements and processing time needed for its implementation. The example above, for instance, might require more than the available memory to implement, and the processing necessary to compute the eligibility tables would probably increase, rather than decrease, the total processing time if only one or two requirements were to be analyzed daily.

The criteria upon which diversion options are selected are historically somewhat dynamic in their details. The system is designed to enable modifications and enhancements to the selection processing, particularly with regard to greater precision in eliminating ineligible missions. The modular structure, and the serial manner in which modules are used to evaluate missions for eligibility make changes to the selection process very simple. For example, when accurate cargo and passenger data becomes available in the AIMS data base, a module to eliminate missions when the cargo capacity of the aircraft would be exceeded can be added to the processing sequence very easily.

The modules which evaluate missions and select diversion options should be periodically reviewed and updated to reflect more current, detailed and accurate judgments on how this should be done. The users' experience in rescheduling missions from the diversion options produced

by the system should provide the basis for these judgments. This periodic review, and modification when warranted, is essential to keeping the system useful, and improvements in the accuracy and detail of the selection criteria will provide a valuable basis for any follow-on system to automate the total rescheduling function more completely.

BIBLIOGRAPHY

1. Air Force Manual 51-40. Air Navigation. Washington, D.C.: Department of the Air Force, October 1959.

2. Bickner, Robert E. Cargo Density and Airlift. Santa Monica, California: The Rand Corporation, January 1957. (AD 112417)

3. Constatine, Larry L. and Edward Yourdan. Structured Design. New York: Yourdan Press, 1975.

4. Dantzig, George B. Linear Programming and Extensions. Santa Monica, California: The Rand Corporation, August 1963. (AD 418366)

5. Dickover, Melvin E. Principles of Coupling and Cohesion for Use in the Practice of SofTech's Structured Analysis and Design Technique. Waltham, Massachusetts: SofTech, Inc., March 1976.

6. Hewlett-Packard. HP9825A Calculator Advanced Programming. Loveland, Colorado: Hewlett-Packard Calculator Division, April 1977.

7. Hewlett-Packard. HP9825A Calculator Disk Programming. Ft Collins, Colorado: Hewlett-Packard Ft Collins Division, September 1977.

8. Hewlett-Packard. HP9825A Calculator General I/O Programming. Loveland, Colorado: Hewlett-Packard Calculator Division, January 1977.

9. Hewlett-Packard. HP9825 Calculator Operating and Programming. Loveland, Colorado: Hewlett-Packard Calculator Division, April 1976.

10. Hewlett-Packard. HP9825 Calculator Systems Programming. Loveland, Coloradl: Hewlett-Packard Calculator Division, August 1977.

11. Hewlett-Packard. String Variable Programming. Loveland, Colorado: Hewlett-Packard Calculator Division, April 1977.

12. Hewlett-Packard. HP98034A HP-IB Interface Installation and Service Manual. Loveland, Colorado: Hewlett-Packard Calculator Division, July 1976.

13. Hewlett-Packard, HP98036A Serial I/O Interface Installation and Service Manual. Loveland, Colorado: Hewlett-Packard Calculator Division, September 1976.

14. Hewlett-Packard. HP9872A Graphics Plotter. San Diego, California: Hewlett-Packard San Diego Division, September 1977.

15. Honeywell. Honeywell Series 60/6000 GCOS Software: TSS System Programmer's Reference Manual. Waltham, Massachusetts: Honeywell Information Systems Inc., March 1974.

16. Maneely, John R. Design of a Laboratory Data Acquisition System (Time Digitization System). MS Thesis. Wright-Patterson AFB, Ohio: Air Force Institute of Technology, March 1978.

17. Military Airlift Command. H6000 Users Guide. Scott AFB, Illinois: Military Airlift Command Deputate for Data Automation, May 1973.

18. Millvish, Robert K. Mathematics of Map Projections. London, UK: Cambridge Press, 1931.

19. Petersen, J.B. Lecture materials distributed in EE6.93, Software Engineering. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1978.

20. Robinson, Arthur H. Elements of Cartography. New York, New York: John Wiley and Sons, Inc., 1960.

21. Sharpe, W.F. A Model for Selecting, Routing and Loading Aircraft. Santa Monica, California: The Rand Corporation, June 1966. (AD 636411)

22. Wiederhold, Gio. Database Design. New York, New York: McGraw Hill, Inc., 1977.

23. 9022-78R. An Introduction to SADT Structured Analysis and Design Technique. Waltham, Massachusetts: SofTech, Inc., November 1976.

APPENDIX A

STRUCTURED ANALYSIS DIAGRAMS (REF 16)

APPENDIX A

STRUCTURED ANALYSIS DIAGRAMS (Ref 16)

This Appendix gives a short description of how Structured Analysis models are constructed and explains the SA diagram conventions used in this paper. It must be noted that the format used to present the models in this paper is not standard according to the rules developed by SofTech. The changes were made to present the models in a manner which is more familiar to readers who have no experience with SA models. Although the format is not that used by SofTech, the diagrams of the models are organized and related according to SofTech procedures, and the conventions used to construct individual diagrams are standard.

The Structured Analysis Design Technique is a general purpose top-down, modular technique for modeling functions. The functions may be as varied as farming or manufacturing, but SA was developed primarily as a software requirements definition and design tool. Although a complete SA model actually consists of two models, one for activities and one for data, this paper employs only activity models so the conventions described here are those which apply to activity models.

An SA activity model consists of a series of diagrams which present in progressively more detail the activities necessary to perform some function. Each diagram represents a self-contained activity which is part of the overall function. A diagram shows how its activity is decomposed into subactivities, and how the subactivities are related to

each other. The subactivities in each diagram may then be decomposed on separate diagrams which leads to a tree structure of several levels. At the top is one diagram which represents the whole function, and at the bottom are the diagrams which show the most detailed activities.

Figure A-1 shows how an SA model would appear if all the diagrams were on one page. Of course, in real SA diagrams only one level of decomposition is shown, but the figure demonstrates the top down nature of SA and the way activities are grouped into modules. In the figure, as in real models, one large box represents the whole function, and that is decomposed into successive levels of related activities. The decomposition process continues until the desired amount of detail has been developed, which may require more levels than shown in Fig A-1. Another thing to note is that while the figure shows only 3 subactivities in each decomposition, any number from 3 to 7 is acceptable.

From Fig A-1, it should be apparent that SA diagrams are constructed with boxes and arrows. In an activity model, each box represents an activity, and is called a node. Arrows represent "data" where the word data is used in a very general sense to include anything that is not an activity. Fig A-2 shows the different meanings given to arrows depending on which side of a box they enter or leave. An input is data that is modified by the activity to produce an output. A control is data which may or may not be converted into output, but which in some way restricts the activity (starts or stops it for example). Every box must have at least one control arrow. A mechanism is a person or thing which acts as a processor. Mechanism arrows are often omitted when the processor

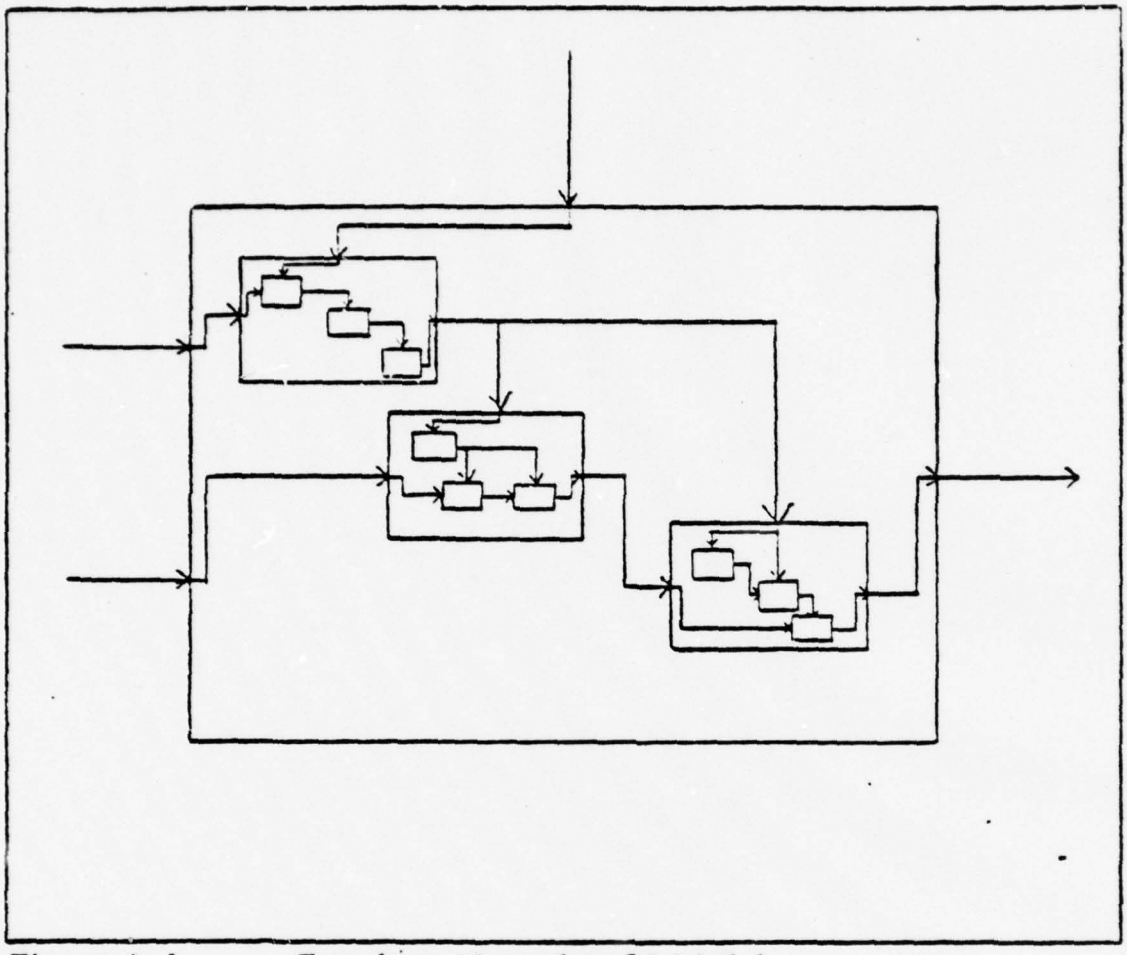


Figure A-1 Top-down View of an SA Model

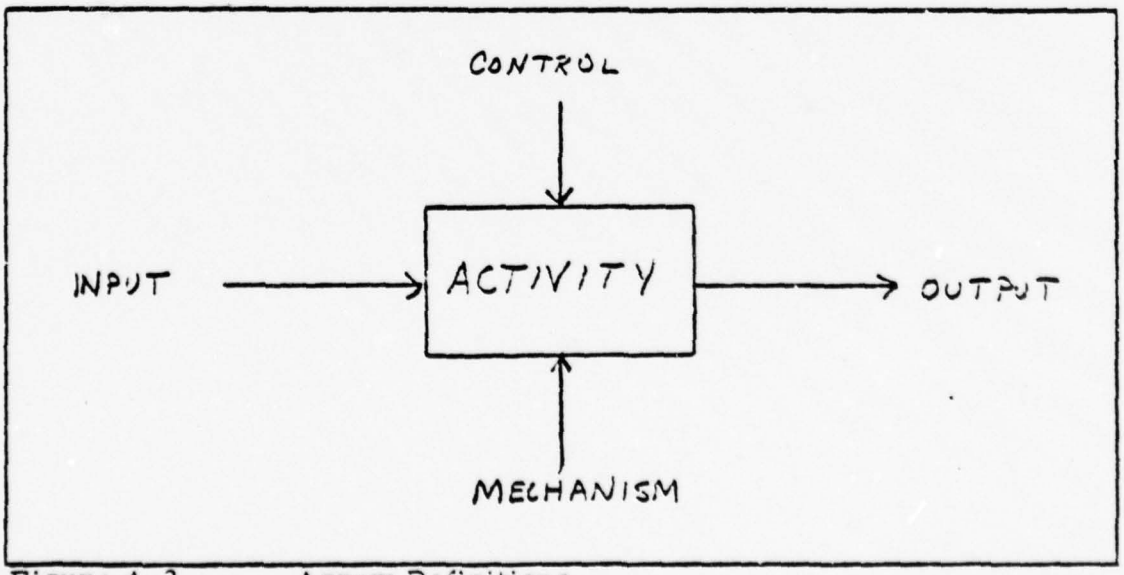


Figure A-2 Arrow Definitions

is the same for all nodes. No limit is placed on the number of arrows which may interface with a side of a box, but it is common practice to group related types of data.

Between boxes, arrows may split and join. In general, all branches of an arrow contain the same data unless a branch is given a separate label. This convention is summarized in Fig A-3 which also gives two forms of OR-branches. The OR-branches are used to show that data follows one path or the other, but not both.

When two nodes are related so that the output of each is a control for the other, a special two-way arrow may be used. Fig A-4 shows a mutual control situation with a two-way arrow and the equivalent form with normal arrows. An arrow showing mutual control has two labels separated by a slash; the first label identified data going forward, and the second is the feedback data.

A special numbering system is used to distinguish between nodes at different levels and between nodes at the same level. In an activity model, node numbers are prefixed with the letter A. For preliminary nodes, A is followed by a dash and a number. Node A-1 may be used to show the model in relationship to other functions (Fig 2-3). Node A-0 serves as a cover sheet for the model; the node is simply a box showing inputs, outputs, controls, and mechanisms for the function which the model is to describe (Fig 2-4). Decomposition begins in Node A0. Note in Fig 2-5 that each box of the decomposition is numbered; the boxes on all decomposition diagrams are numbered, and this number is used to form the node number. For the activities subordinate to Node A0, the

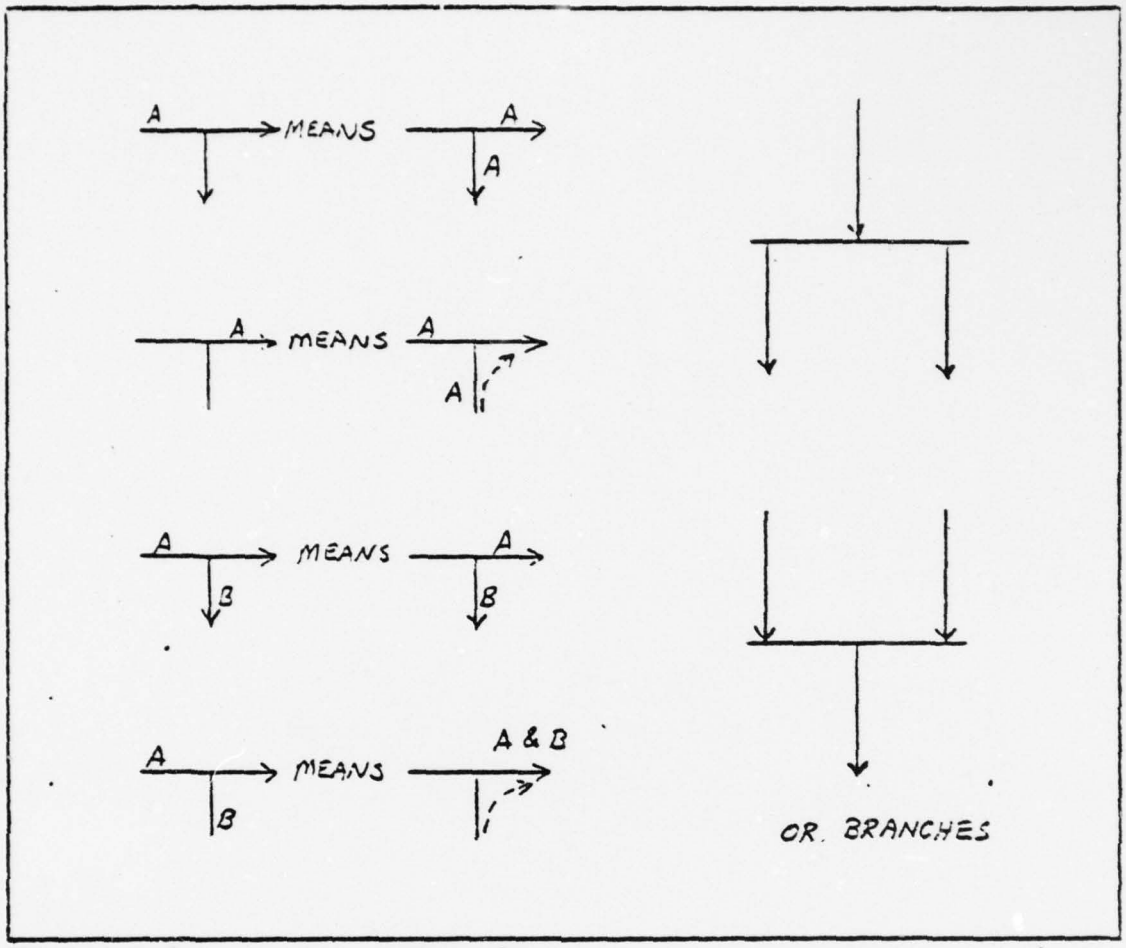


Figure A-3 Arrow Branches

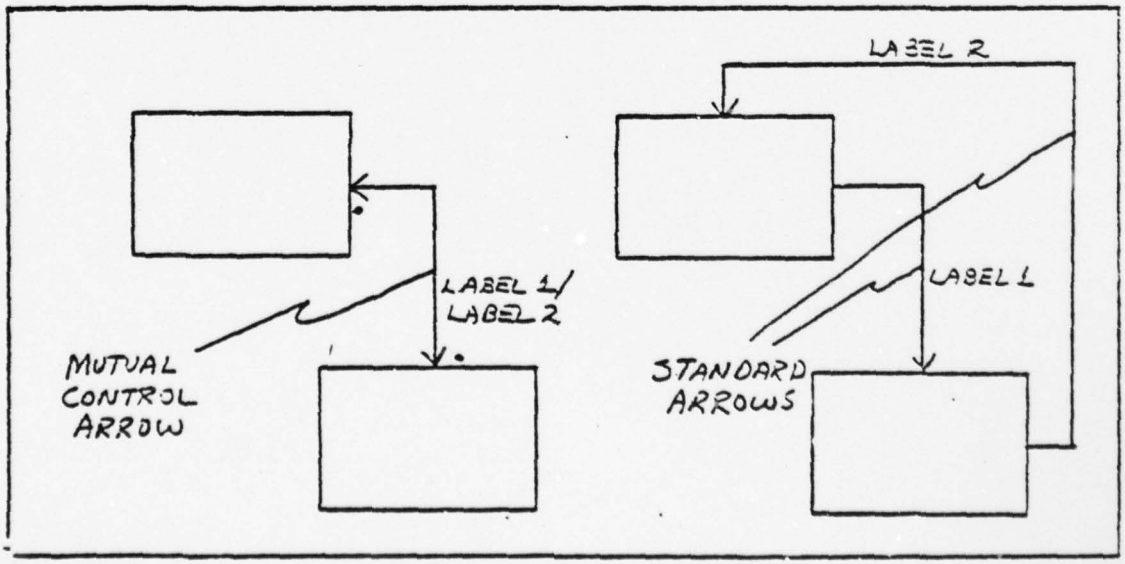


Figure A-4 Arrows Showing Mutual Control

node number is simply the box number on A0; Selection Diversion Eligible missions in Fig 2-6, for example, becomes Node A1. From this level on, the node number is a combination of the node number of the parent diagram and the box number of the subordinate. As an example, the decomposition of Select Diversion-Eligible Missions is given in Fig 2-6. Box 2, Reconstruct Mission Itinerary, is assigned the node number A12. Subordinates of A12, if diagrammed, would have the numbers A121, A122, and so on through the last box number.

A special code called an ICOM code (Input, Control, Output, Mechanism) is used to identify arrows. The code contains a number, a letter, and another number. The first number is that of the box which the arrow enters or departs. The letter refers to the type of arrow, I for input, C for control, O for output, and M for mechanism. The last number distinguishes between arrows of the same type on a box; numbers are assigned counting from left to right and top to bottom. In Fig 206, the code 5I2 would refer to the arrow labeled Integrated Airlift Schedule Data which is an input to Extract/Store Data, box 5. It should be apparent that an ICOM code is not unique because arrows can be connected to several boxes. Codes used in the text are derived from the box which is of the most interest in the discussion.

On a diagram, arrows which enter or depart the entire diagram are given a shortened version of the ICOM code. In the shortened version, the box number is omitted and the remainder of the code refers to the position of the same arrow on the parent diagram. For example, the output Diversion Eligible Mission Data in Fig 2-6 is coded O1 because

Diversion-Eligible Mission data is the first output from box 1 of Node A0 in Fig 2-5 which is the parent diagram of Node A1.

Two important points about the text describing each SA diagram must be included here. The text is intended to point out the highlights of a diagram and not repeat all the details. As an aid to following the discussion, both the long and short versions of the ICOM code as well as box numbers are used as a reference to specific diagram features.

APPENDIX B

MODULE RELATEDNESS MEASUREMENT (Ref 19)

APPENDIX B

MODULE RELATEDNESS MEASUREMENT (Ref 19)

COUPLING - A measure of the intermodule strength of connection - minimize the relationship among modules.

COHESION - A measure of the intramodule strength - maximize the module strength.

MODULE COUPLING

DATA	LOWEST (BEST)
STAMP	
CONTROL	
EXTERNAL	
COMMON	
CONTENT	HIGHEST (WORST)

MODULE COHESIVENESS

FUNCTIONAL	HIGHEST (BEST)
CLUSTERED	
SEQUENTIAL	
COMMUNICATION	
PROCEDURAL	
TEMPORAL	
LOGICAL	
COINCIDENTAL	LOWEST (WORST)

COHESION LEVELS

1.	FUNCTIONAL:	Module performs a single function.
2.	CLUSTERED:	Module is a group of functions sharing a data structure; only one function is performed per invocation.
3.	SEQUENTIAL:	Module action comprises several functions that pass the data along.
4.	COMMUNICATIONAL:	Module action consists of several logical functions operating on some data.
5.	PROCEDURAL:	Module elements are grouped for algorithmic reasons.
6.	TEMPORAL:	Module functions are all related in time (e.g., initialize)
7.	LOGICAL:	An invoking parameter value determines the specific function to be performed of a general function.
8.	COINCIDENTAL:	No real relationship between module elements that are grouped for packaging considerations.

COUPLING LEVELS

DATA:	All communication between modules is via arguments that are data elements.
STAMP:	Communication includes an argument that references a data structure; sóme of whose fields are not needed.
CONTROL:	An argument from one module knowingly influences the flow-of-control of the other (e.g., flat).
EXTERNAL:	Modules reference an externally declared individual data element.
COMMON:	Modules reference an externally declared data structure (e.g., common).
CONTENT:	One module references the contents of the other.

APPENDIX C

HEWLETT PACKARD LANGUAGE (HPL) NOTATIONS

(Refs 6 through 11)

APPENDIX C

HEWLETT PACKARD LANGUAGE (HPL) NOTATIONS

(Refs 6 through 11)

VARIABLES

r - variables rØ,r1,...r(n) Global Variables

A,B,C,...Z Global Variables

p - variables p1,p2,...p(n) Subroutine local variables

Arrays: A[row, column],...A[row,column] Arrayed Variables

Strings: A$[1st Byte,last Byte],...A$[1st byte,last byte] -

Row-relational data allocations in byte (8 bits) Format

String Arrays: A$[row #, 1st byte, last byte],...Z$[row #, 1st byte,

last byte]- An array of data in string format.

OPERATIONS

→ Assignment Operator; e.g., 10 → A (Value of 10 is assigned to A)

+ - * / Add, subtract, multiply, divide operators

↑ √ Exponential and Square Root operators

= < > Relational Operators

FUNCTIONS

log, sin, tan, mod, abs, max,....etc. Equivalent to corresponding FORTRAN intrinsic functions.

LABELS

"labelname": or line numbers may be used as absolute program addresses.

PROGRAM BRANCHING

Branching is accomplished similarly to FORTRAN; e.g.:

ABSOLUTE BRANCHING: gto "labelname" or gto line #

RELATIVE BRANCHING: jmp + #lines or jmp - #lines

SUBROUTINE CALLS: gsb "subroutinename" or cll 'subroutinename' (variable #1, variable #2,...variable #n).

INSTRUCTION SET

Read Only Memory cards are used by the 9825A calculator to extend the basic calculator instruction set to include read/write I/O operations to all peripherals, special instruction routines for the plotter and disk, and other internal operations which will execute like intrinsic functions.

DATA STORAGE

Numerical variables may be stored with full precision, by normal assignment operations, into 8 bytes of internal memory; split precision is stored into 4 bytes and integer precision into 2 bytes. Split or integer format are stored into string variables.

Notation:		
	full to split	fts(A) → B$[1,4]
	split to full	stf(B$[1,4]) → A
	full to integer	fti(A) → B$[5,6]
	integer to full	itf(B$[5,6]) → A

DATA I/O

Format statements, applicable to read and write operations, are essentially similar to FORTRAN.

PERIPHERAL ADDRESSES

The following peripherals used by the system have been assigned addresses as follows:

CRT: 10

Serial Interface to WWMCCS Mainframe = 11

Plotter = 705

Printer = 701

Internal Printer = 16

DISK STORAGE

Both program/program segments and data may be stored on disk. Data may be stored as strings or as full precision variables. Data or programs are stored on named files on the disk: programs are automatically allocated sufficient disk storage when saved, while the file size for data must be assigned in numbers of records. Disk records are 256 bytes in length. Both random and sequential access to disk data files are possible. Random read and write operations must specify the file and record number; serial operations occur sequentially from the file pointer, reading or writing the number of bytes specified. The file pointer remains at the end of the last byte read or written unless repositioned at the beginning of a file or record by a random read operation.

APPENDIX D

INQUEST PROGRAM TO SELECT DIVERSION ELIGIBLE MISSIONS

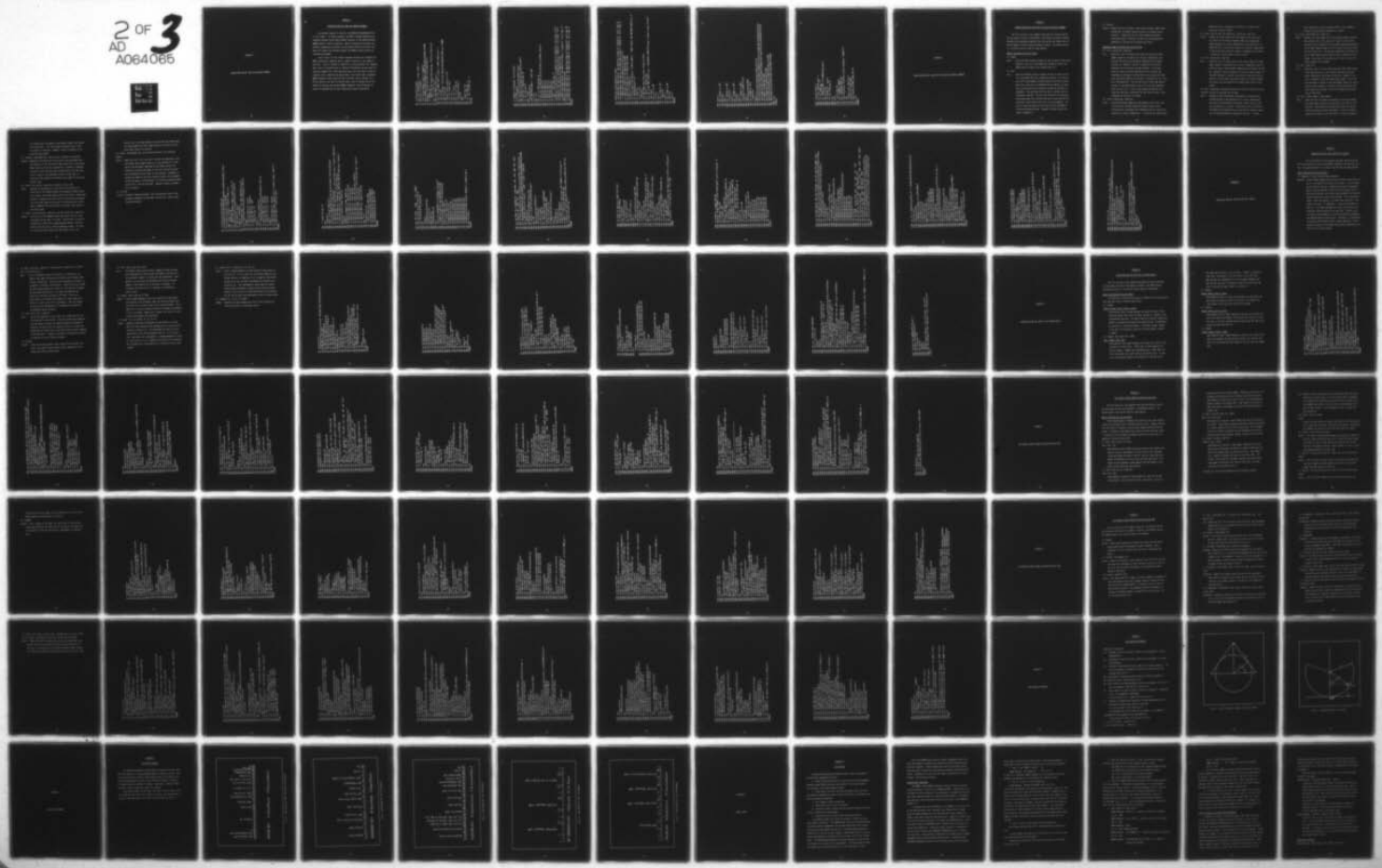
```
0010##STRIP NORMAL
0020$       IDENT   FA029    /1 /AMCC      ,ZZZ/2DOOMO/           ,F G101NQ
0030*     JLIST 20,110,610
0040$       SELECT  DOOMA/FAT/JCL03A
0050$       SELECT  DOAIMS/A005/ZBN4WWQJU
0060$       FILE    OR,X1S
0070$       DATA    DC
0080SELECT((60 = "J" OR "V") OR (60 = "EN" OR "UN" AND 64 = "09"
0090SELECTOR "10" OR "05" OR "06" OR "L3")) AND 94 - "Y" AND ((21 =
0100SELECT"005" OR "130" OR "141") OR (22 = "130")) OR (61 = "M")
0110SELECTAND 256 = "269" TO "276"
0120TABL0160, "J" = "DE", "V" = "DE", D = "  "
0130TABL02272, "C" = "R", D = " "
0140RCDOA L13,"      ",59/12,21/3,249/4,259/4,256/3,268/4,278/4
0150RCDOA 275/3,97/3,"   ",T01/2,"    ",T02/1,1/12,263/5,282/5
0160$:SELECT:DOOMA/FAT/JCL03A
0170$:FILE:IS,X1R
0180$:FILE:OR,X2S
0190$:DATA:DC
0200SORT  57,12,A
0210SIZES 000013000013
0220CTL   B1 = 57/12
0230CNTRS C01 = X
0240RCDOA L13,1/4,R1A/2,7/72
0250RCD1AZL01,"999999"
0260$:SELECT:DOOMA/FAT/JCL03A
0270$:FILE:IS,X2R
0280$:FILE:OR,X3S
0290$:DATA:DC
0300SORT  57,12,A
0310SIZES 000013000013
```

```
0320SELECT1="999999"
0330CTL    B1 = 57/12
0340IF01   5 = "01"
0350CNTRS C01 = X IF01
0360RCD0A L13,R01A/4,5/74
0370RCD1A L01,"999999"
0380$:SELECT:D00MA/FAT/JCL03A
0390$:FILE:IS,X3R
0400$:FILE:OR,X4S
0410$:DATA:DC
0420SORT  57,12,A
0430SIZES 000013000013
0440SELECT1="999999"
0450CTL    B1 = 57/12
0460CNTRS C01 = X, C02 = 69$3, C03 = 74$3, C04 = 72$2
0470CNTRS C05 = 7/$2
0480COMPUTC06 = C02 * 60
0490COMPUTC07 = C03 * 60
0500COMPUTC08 = C06 + C07 + C04 + C05
0510COMPUTC10 = C08 / 6
0520RCD0A L10,1/51,R10A/4,56/1
0530RCD1AZL01,"999999"
0540$:SELECT:D00MA/FAT/JCL03A
0550$:FILE:IN,X4S
0560$:PRMFL:OR,R/W,S,D00MA/FAT/HPSCED
0570$:DATA:DC
0580SELECT1 - "999999"
0590RCD0A L10,"&",1/56
0600$:ENDJOB
```

AD-A064 065 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 15/5
OPPORTUNE RESCHEDULING SYSTEM FOR MILITARY AIRLIFT COMMAND CARG--ETC(U)
DEC 78 G F SANDERSON

UNCLASSIFIED AFIT/GCS/EE/78-17 NL

2 OF 3
AD
A064065




OF
3
4065

APPENDIX E

DESCRIPTION AND HPL CODE FOR TRANSFER PROGRAM

APPENDIX E

DESCRIPTION AND HPL CODE FOR TRANSFER PROGRAM

The transfer program is stored on the OPPORTUNE RESCHEDULING disk in File "trans". To effect transfer, the WWMCCS teletype motem must be connected through the MIL-188-to-RS232C converter to the Hewlett-Packard 98036A (Option 1) Serial Interface. When the connections are made, the transfer program may be started, and the Hewlett-Packard calculator keyboard will emulate the teletype keyboard for WWMCCS system sign-on and time-sharing commands.

Lines 10 and 11 of the transfer program configure the interface for WWMCCS conventions regarding parity, number of data bits, and number of stop bits. Lines 13 through 17 establish an interrupt buffer for incoming data; lines 31 through 60 and 111 through 119 establish the two levels of interrupt needed to put individual data bytes into the buffer as they are received, and to identify and process data in the buffer when a complete WWMCCS message or data record has been received; lines 4 through 9 initialize space on the disk for mission data; lines 18 through 30 and 102 through 110 display and transmit WWMCCS commands; lines 65 through 101 convert HP keycoded data to ASCII coded bytes before transmission.

```
0: "trans-trk1,file4":
1: ent "CRT CONNECTED?  1=YES,  0=NO",r1
2: 16→r0; if r1=1; 10→r0
3: lkd ;files mens; on end 1,"nospace"
4: dim L$[98],C[255],D$[98],A$[80],M$[256],B$[80],P$[80]
5: 0→R; ent "INITIALIZE MSN FILE? 999=YES",R
6: if R#999; jmp 5
7: " "→P$[1,41]; "0"→P$[42,42]; 0→R; dsp "INITIALIZING FILE"
8: for I=1 to 211 by 42
9: P$[1,42]→M$[I,I+41]; next I
10: for I=1 to 200; rread 1,I; sprt 1,M$[1,252]; next I
11: wsm 11,251
12: wso 11,37; dsp "TELETYPE EMULATION BEGINS NOW"; wait 3000
13: gsb "string"
14: "setup input buffer"; 1→Z→H
15: oni 11,"interrupt"
16: eir 11,4
17: buf "in",L$,1
18: tfr 11,"in",82,13
19: "keyboard output":
20: wti 0,0
21: 0→P
22: "loop": rdi 4→K
23: if rdi 4#K; jmp -1
24: if (rdi 4→C)#K; jmp -2
*17157
```

```
25: if C=66 or C=194; gsb "controlchar"
26: if C=123; jmp 3
27: wtb 11, C[C]→P; char(P)→A$[S+1→S, S]
28: dsp A$[max(1, len(A$)-31), max(32, len(A$))]
29: if C[C]=13; ""→A$; 0→S; dsp A$
30: if rdi 4=C; jmp 0
31: gto "loop"
32: "interrupt"; 0→J
33: L$[1, 80]→P$[1, 80]; " "→L$[1, 82]
34: cir 11, 4
35: buf "in"
36: tfr 11, "in", 82, 13; efg 14
37: if H=1; conv 13, 32; wtb r0, 10, 13, P$[1, len(P$)], 13; conv
38: if P$[1, 2]="&0"; 1→J
39: if P$[2, 3]="&0"; 2→J
40: if P$[3, 4]="&0"; 3→J
41: if P$[4, 5]="&0"; 4→J
42: if P$[5, 6]="&0"; 5→J
43: if J=0; " "→P$[1, 80]; jmp 17
44: val(P$[J+1→J, J+3])→F; fti (F)→M$[Z, Z+1]; if flg15; oll 'prt'; gto "intr"
45: val(P$[J+4, J+5])→G; fti (G)→M$[Z+2, Z+3]; if flg15; oll 'prt'; gto "intr"
46: "*"→M$[Z+4, Z+4]; P$[J+6, J+17]→M$[Z+5, Z+16]
47: val(P$[J+18, J+20])→Q; fti (Q)→M$[Z+17, Z+18]; if flg15; oll 'prt'; jmp 14
48: P$[J+21, J+24]→M$[Z+19, Z+22]
49: val(P$[J+25, J+28])→Q; fti (Q)→M$[Z+23, Z+24]; if flg15; oll 'prt'; jmp 12
50: val(P$[J+29, J+31])→Q; fti (Q)→M$[Z+25, Z+26]; if flg15; oll 'prt'; jmp 11
*32723
```

```
51: P$[J+32,J+35]→M$[Z+27,Z+30]
52: val(P$[J+36,J+39])→Q;fti (Q)→M$[Z+31,Z+32];if flg15;oll 'prt';jmp 9
53: val(P$[J+40,J+42])→Q;fti (Q)→M$[Z+33,Z+34];if flg15;oll 'prt';jmp 8
54: " "→M$[Z+35,Z+40];"@"→M$[Z+41,Z+41];ofg 14
55: if P$[J+49,J+49]="D";"D"→M$[Z+39,Z+39]
56: if P$[J+55,J+55]="R";"R"→M$[Z+40,Z+40]
57: Z+42→Z;if Z>=252;1→Z
58: if Z=1;rread 1,R+1→R;eprt 1,M$[1,252];" "→M$[1,251];"@"→M$[252,252];prt R
59: if r0=16;0→H;prt F,G
60: if J=0;1→H
61: "intr";iret
62: "nospace":
63: wrt 16,"MSN FILE FULL";fmt f4.0,o1,f3.0
64: wrt 16,C," ",D;fmt o14
65: wrt 16,"LAST MSN INPUT";prt "TRANSFER COMPLETE";gto "setup input buffer"
66: "string":
67: for I=1 to 58
68: I→C[I]
69: next I
70: for I=78 to 87
71: I-30→C[I]
72: next I
73: for I=88 to 96
74: 32→C[I]
75: next I
*8338
```

```
76: next I
77: for I=123 to 175
78: 32→C[I]
79: next I
80: for I=176 to 185
81: I-144→C[I]
82: next I
83: for I=186 to 224
84: 32→C[I]
85: next I
86: for I=225 to 250
87: num(chr(I-160))→C[I]
88: next I
89: 13→C[141]; 10→C[138]
90: for I=206 to 216
91: I-158→C[I]
92: next I
93: 60→C[172]; 123→C[123]; 94→C[94]; 32→C[7]
94: 47→C[47]→C[175]; 40→C[40]→C[168]; 41→C[41]→C[169]; 92→C[222]; 62→C[174]
95: 101→C[96]→C[224]; 125→C[125]→C[253]; 43→C[43]→C[171]; 45→C[45]→C[173]
96: 27→C[65]→C[193]; 63→C[63]
97: 64→C[183]; 91→C[184]; 93→C[185]
98: 39→C[176]; 8→C[20]→C[148]; 61→C[61]→C[189]
99: 59→C[59]→C[187]; 58→C[191]; 46→C[88]→C[216]
100: ret
*28454
```

```
101: "controlchar":
102: if (rdi 4+M)=C: jmp -1
103: if rdi 4#M: jmp -1
104: if (C[M]+C)<64 or C>90: beep: jmp 2
105: wtb 11,C-64
106: if C=122: oll 'clear'
107: if rdi 4=M: jmp 0
108: ret
109: end
110: "prt": ofg 15
111: prt "FORMAT ERROR"
112: prt "MISSION SEQ NR."
113: fmt 2f4.0: prt F,G: fmt c4,2x,f3.0: prt "REC#",R
114: ret
115: "clear": prt "Buf Clr"
116: bred("in")+P$[1,80]
117: wtb r0,10,13,P$[1,40],10,13
118: ret
*20871
```

APPENDIX F

DESCRIPTION AND HPL CODE FOR THE EXECUTIVE CONTROL PROGRAM

APPENDIX F

DESDRIPTION AND HPL CODE FOR THE EXECUTIVE CONTROL PROGRAM

The first section of this Appendix describes the function and HPL calling syntax for each of the modules in the Executive Control Program, and each of the procedural modules which are stored, and loaded into calculator memory, with the Executive Control Program. The second section is a listing of the HPL code for these modules.

MODULE FUNCTIONS AND CALLING SYNTAX

Cll 'enter'

enter - Loads the ENTER overlay; prompts the user to enter a data source selection, then calls the appropriate subroutine within the ENTER overlay to modify the selected data file.

Cll 'opt'

opt - Loads the OPTIMIZE overlay; prompts the user to select an airlift requirement which has already been stored, or to enter a new airlift requirement; and then analyzes the entire mission data file, using calls to OPTIMIZE overlay modules, to select the six best missions for diversion to meet the selected requirement. The resulting diversion data is stored, and also displayed on the CRT. The user has the option to re-analyze the mission data file to select the next six-best options for diversion in case none of the first six are acceptable. The user is given the option to print and plot the requirement, mission and diversion data. Examples of these displays are shown in Appendix N.

C11 'display'

display - Prompts the user to select a data source, and then loads either the MAP PLOT or DISPLAY overlay according to the data source selection. Prompts the user to select whether plotting or printing the data is desired, and then calls the appropriate subroutine in the overlay to perform the display.

PROCEDURAL MODULE FUNCTIONS AND CALLING SYNTAX

C11 'icrt' (record nbr, index nbr)

icrt - Reads the "icao" file data stored under the record and index number passed to the module as p1 and p2, respectively, and displays the data on the peripheral addressed by the global variable rØ. Normally, the CRT is addressed by rØ unless a hardcopy print of the data has been selected. The display shows the ICAO four-letter identifier for an airfield, the longitude and latitude of the airfield, and a three digit code which identifies whether the airfield is suitable for C-5, C-141 and C-130 operations. The left-most digit of the code applies to the C-5; the middle digit to the C-141, and the right-most digit to the C-130. A zero in the respective position indicates the airfield is unsuitable for that type aircraft; a one indicates the airfield is suitable.

C11 'ifind' (record nbr, index nbr)

ifind - Locates the record number and index number of the "icao" data file entry for the ICAO identifier letters stored in the string L$[1,4], and passes this information back to the calling routine as p1 and p2 respectively. If data for the ICAO letters

cannot be found, a message to this effect is printed on the calculator internal printer.

Cll 'mfind' (msn seq nbr, msn subseq nbr, record nbr, index nbr)

mfind - Locates the record number and index number of the "msn" data file entry for the mission sequence number and subsequence number passed as p1 and p2, respectively, and returns this information back to the calling routine as p3 and p4, respectively. If the mission data cannot be located, a message to that effect is printed on the calculator internal printer.

Cll 'mcrt' (record nbr, index nbr)

mcrt - Reads the "msn" file data stored in the record number and index number passed to the module as p1 and p2, respectively, and displays the data on the peripheral addressed by the global variable rØ. Appendix N shows an example of the mission data display. All the data displayed is stored in the mission record with the exception of flying time; subroutine "mtime" is called to compute the flying time from take-off and landing times stored in the record.

Cll 'proj' (longitude, latitude) Horizontal and vertical plotter units are returned in place of longitude and latitude.

proj - Computes plotter coordinates corresponding to longitude and latitude passed to the subroutine as p1 and p2, respectively, and returns the horizontal and vertical plotter units to the calling routine as p1 and p2. The plotter units are scaled to correspond to one Nautical Mile on the earth's surface, regardless of the map boundaries selected by the user. A simple

conic projection is used to compute plotter units; Appendix K explains the projection mathematics in detail.

C11 'rfind' (requirement nbr, index nbr)

rfind - Computes the index number for the requirement number passed to the subroutine as p1 and returns this value to the calling routine as p2; or computes the requirement number corresponding to the index number passed as p2, and returns the value to the calling routine as p1. The two-way computation facilitates the use of other modules where either or both of these values are needed. Since there is only one record in the "req" data file, record numbers are not needed.

C11 'rcrt' (index nbr)

rcrt - Reads the "req" file data stored under the index number passed to the subroutine as p1, and displays the data on the peripheral addressed by the global variable rØ. Appendix N contains an example of the requirement data display. All of the data displayed is stored in the requirement record with the exception of the distance in nautical miles from onload to offload; subroutine "reqtime" is called to compute the distance from airfield data.

C11 'dcrt' (record number, index number)

dcrt - Reads the "div" file data corresponding to the record number and index number passed to the subroutine as p1 and p2, respectively. One record of diversion data stores six mission diversion options for each requirement. Since nine requirements may be stored in the "req" data file, the nine records

of diversion data correspond to requirement numbers one through nine respectively. The index number determines which of the six options is displayed. Appendix N shows an example of the diversion data display.

C11 'reqtime' (requirement nbr, type aircraft, distance, flying time)

reqtime - Computes the distance and flying time for the requirement number passed as p1 and the aircraft type passed as p2, and returns these values as p3 and p4, respectively. Distance is computed from the airfield location data corresponding to the ICAO identifiers stored in the requirement record; flying time is a function of the distance and average flying speed for the given aircraft type.

C11 'mtime' (msn seq nbr, subseq nbr, distance, flying time)

mtime - Computes the distance and flying time for the mission leg identified by the sequence number and subsequence number passed as p1 and p2, and returns these values as p3 and p4, respectively. Distance is computed from the airfield location data corresponding to the ICAO identifiers stored in the mission data; flying time is computed from the take-off and landing times stored in the mission data.

C11 'mcheck' (msn record nbr, index nbr, last msn record nbr, index nbr)

mcheck - A Military Airlift Command mission may consist of one or more individual mission legs, or flights. Mission data is stored in chronological order with a common sequence number; individual mission legs are given a unique subsequence number. This module finds the record number and index number of the last

mission leg in the same sequence as the mission leg stored under the record number and index number passed as p1 and p2, and returns these values as p3 and p4.

C11 'optcrt' (requirement nbr, first diversion option, last diversion option)

optcrt - Reads the "req", "div" and "msn" file data corresponding to the requirement index number passed as p1, and displays all of the data on the peripheral addressed by the global variable rØ. Parameter p2 selects the number of the first diversion option, and corresponding mission data, to be displayed. Parameter p3 selects the number of the last diversion option, and corresponding mission data, to be displayed. All diversion options between p2 and p3 will also be displayed. Appendix N shows an example of this display.

C11 'crtline'

crtline - Displays a dashed horizontal line, 80 characters long, at the peripheral addressed by the global variable rØ. Used to separate data displays.

```
0: "control": 0→r29→r24; 240→r40
1: dim C$[252], L$[128], M$[252], R$[252], I$[252], D$[252], F[3,5]
2: files map,*,req,div,ioao; 10→r0
3: asgn "mene",2; chain "onefil",240,oln+1
4: oll 'mapsetup'(0)
5: ent "START MAP OVERLAY NOW? Yes/No", L$[1,2]
6: cap(L$[1,1])→L$[1,1]; cap(L$[2,2])→L$[2,2]
7: if L$[1,2]#"YE"; jmp 3
8: 1→r24; oll 'mapintplt'
9: if r24=1; dsp "WAIT TILL MAP COMPLETE"; jmp 0
10: ent "FUNCTION: OPtimize, ENter, DIsplay", L$[1,2]
11: "space":
12: if L$[1,2]="EN"; oll 'enter'
13: if L$[1,2]="DI"; oll 'display'
14: if L$[1,2]="OP"; oll 'opt'
15: " "→L$[1,2]; jmp -5
16: "enter":
17: chain "twofil",240,oln+1
18: ent "MSne, REqmte, ICao, COrds, SPec, DOne", L$[1,2]
19: if L$[1,2]="MS"; oll 'oenmen'
20: if L$[1,2]="RE"; oll 'oenreq'
21: if L$[1,2]="IC"; oll 'oenica'
22: "space":
23: if L$[1,2]="CO"; oll 'oencor'
24: if L$[1,2]="SP"; asgn "spec",2; oll 'oenspc'; asgn "mene",2
25: if L$[1,2]="DO"; " "→L$[1,2]; jmp 2
*22043
```

```
26: " "→L$[1,2]; jmp -8
27: ret
28: "display":
29: chain "trefil",240,oln+1
30: "die": ent "MS,MI,RE,CO,IC,SP,PE,MA,OP,DOne",L$[1,2];0→p3→p4
31: if L$[1,2]="DO"; gto "displayret"
32: if L$[1,2]="MA" or L$[1,2]="PE" or L$[1,2]="CO"; chain "onefil",240,oln+1
33: if L$[1,2]="MA"; oll 'mapsetup'(1); gto "display"
34: if L$[1,2]="PE"; oll 'pens'(r22,r23,r25,r26,r27,r28); gto "display"
35: 0→p1; ent "PLOT? 1=YES, 0=NO",p1
36: "space":
37: 0→p2; ent "PRINT? 1=YES, 0=NO",p2
38: if L$[1,2]="MS"; oll 'mdieall'(p1,p2)
39: if L$[1,2]="RE"; oll 'rdie'(p1,p2)
40: if L$[1,2]="IC"; oll 'idie'(p1,p2)
41: if L$[1,2]="CO"; oll 'odie'(p1,p2); gto "display"
42: if L$[1,2]="SP"; asgn "spec",2; oll 'mdie'(p1,p2); asgn "mens",2
43: if L$[1,2]="MI"; oll 'mdie'(p1,p2)
44: if L$[1,2]="OP"; oll 'optdie'(p1,p2)
45: gto "die"
46: "displayret"; ret
47: "iort"; rread 5,p1,I$[1,252]
48: itf(I$[p2+4,p2+5])→p3; p3/100→p3
49: itf(I$[p2+6,p2+7])→p4; p4/100→p4
50: itf(I$[p2+8,p2+9])→p5; fmt o27
*29100
```

```
51: wrt r0,"ICAO    LONG      LAT  CODES"
52: fmt o4,1x,f7.2,1x,f7.2,2x,f4.0
53: wrt r0,I$[p2,p2+3],p3,p4,p5
54: ret
55: "proj":
56: r13+3438*tan(r12-p2)→p3
57: r16*(p1-r11)→p2
58: p3sin(p2)→p1
59: r13-p3cos(p2)→p2
60: ret
61: "mort":if p2mod42#1 or p1=0;gto "mortret"
62: if p3#0 and p3>=p1 and p4mod42=1;1→p16
63: p1→p17;p2→p18;if p1=p3 and p2=p4;0→p16
64: rread 2,p1,M$[1,252]
65: oll 'mtime'(p2,p14,p15)
66: itf(M$[p2,p2+1])→p5
67: itf(M$[p2+2,p2+3])→p6
68: itf(M$[p2+17,p2+18])→p7
69: itf(M$[p2+23,p2+24])→p8
70: itf(M$[p2+25,p2+26])→p9
71: itf(M$[p2+31,p2+32])→p10
72: itf(M$[p2+33,p2+34])→p11
73: itf(M$[p2+35,p2+36])→p12
74: itf(M$[p2+37,p2+38])→p13;if p2#p18 or p1#p17;jmp 4
75: fmt 3x,o56,z
*30518
```

```
76: wrt r0,"SEQ  NO  MISSION ID  TYPE DEPT  DAY  ETD  ARR  DAY  ETA"
77: fmt o16;wrt r0," CGO  PAX  HRS"
78: fmt 4x,f3.0,1x,f3.0,1x,o12,1x,f4.0,1x,o4,1x,f4.0,1x,f4.0,1x,o4,z
79: wrt r0,p5,p6,M$[p2+5,p2+16],p7,M$[p2+19,p2+22],p9,p8,M$[p2+27,p2+30]
80: fmt x,4f4.0,x,f5.1;wrt r0,p11," ",p10," ",p12," ",p13,p14
81: if p16=0;jmp 4
82: p2+42→p2;if p2>=253;p1+1→p1;1→p2
83: if p1>p3 or p1=p3 and p2>p4;0→p16;jmp 2
84: jmp -20
85: "mcrtret";p17→p1;p18→p2;ret
86: "mfind";0→p3→p4
87: on end 2,"nomen"
88: fti (p1)→L$[1,2]
89: fti (p2)→L$[3,4];dsp "SEARCHING"
90: rread 2,p3+1→p3,M$[1,252]
91: for R=1 to 211 by 42
92: if M$[R,R+3]=L$[1,4];R→p4;jmp 3
93: next R
94: if p4=0;jmp -4
95: "mfindret";ret
96: "nomen";p6+1→p6;0→p3
97: if p6=1;fmt 2f3.0;wrt 16,p1,p2," NOT FOUND";if p5=1;jmp -2
98: if p6=1;" "→L$[1,41];"0"→L$[42,42]
99: if p6=1;rread 2,p3+1→p3;eread 2,M$[1,252];pos(M$,L$[1,42])→p4
100: if p4=0 and p6=1;jmp -1
*10200
```

```
101: if p4=0 and p6>1;prt "MSN FILE FULL"
102: if p4=0 and p6>1;0→p3
103: gto "mfindret"
104: "ifind";0→p1→p4;4→p5
105: on end 5,"noetation"
106: for I=1 to 4
107: cap(L$[I,I])→L$[I,I]
108: next I
109: "ilook";rread 5,p1+1→p1,I$[1,252]
110: pos(I$,L$[1,p5])→p2;if p2#0 and p4=0;1→p3
111: if p2=0;gto "ilook"
112: "ifindret";ret
113: "noetation";p4+1→p4
114: prt L$[1,4]," NOT FOUND";if p3=1;0→p3;jmp -2
115: 0→p1;" "→L$[1,9];"0"→L$[10,10];10→p5
116: if p4=1 and p3=0;gto "ilook"
117: if p4>1 and p2=0;prt "ICAO FILE FULL";0→p3
118: gto "ifindret"
119: "opt";0→p1→p2→p3
120: chain "forfil",240,oln+1
121: ent "NEW OPTIMIZE FACTORS? 1=YES,0=NO",p1
122: if p1#1 and p1#0;jmp -1
123: oll 'weight'(p1);0→p1
124: ent "REQ#(ENTER 0 to Input New Req)",p2;if p2<0 or p2>9;jmp 0
125: if p2>0;oll 'rfind'(p2,p4);oll 'rort'(p4)
*3290
```

```
126: if p2=0;oll 'reqget'(p2,p4)
127: 0→p1;ent "RIGHT REQMT? 1=YES 2=NO 0=QUIT",p1;if p1=2;jmp -3
128: if p1=0;gto "optret"
129: oll 'diveet'(p4)
130: 0→p1;oll 'optreq'(p1,p4)
131: ent "DISPLAY RESULT=1,RECOMP=2,DONE=0",p3
132: if p3=1;chain "trefil",240,oln+1
133: if p3=1;0→p5;ent "PLOT? 1=YES 0=NO",p5
134: if p3=1;0→p6;ent "PRINT? 1=YES  0=NO",p6
135: if p3=1;oll 'optdie'(p5,p6,p7,p2)
136: if p3=2;oll 'optreq'(p1+1→p1,p4);jmp -3
137: "optret":ret
138: "rort":rread 3,1,R$
139: if p1=0;1→p1
140: if p1mod28#1;gto "rortret"
141: oll 'reqtime'(p1,p11,p10,p12)
142: itf(R$[p1,p1+1])→p2
143: itf(R$[p1+6,p1+7])→p3
144: itf(R$[p1+8,p1+9])→p4
145: itf(R$[p1+14,p1+15])→p5
146: itf(R$[p1+16,p1+17])→p6
147: itf(R$[p1+18,p1+19])→p7
148: itf(R$[p1+20,p1+21])→p8
149: itf(R$[p1+22,p1+23])→p9
150: fmt 11x,o56
*12543
```

```
151: wrt r0,"SEQ ONLOAD  NET DAY OFFLOAD  NLT DAY CGO PAX CODE  N.M."
152: fmt 13x,f2.0,3x,o4,1x,f4.0,1x,f3.0,4x,o4,1x,f4.0,1x,f3.0,1x,f3.0,z
153: wrt r0,p2,R$[p1+2,p1+5],p3,p4,R$[p1+10,p1+13],p5,p6,p7
154: fmt 1x,f3.0,2x,f3.0,1x,f5.0
155: wrt r0,p8,p9,p10
156: "rortret":ret
157: "dort":if p1<1 or p1>9;gto "dortret"
158: if p2mod42#1;gto "dortret"
159: rread 4,p1,D$[1,252];etf(D$[p2,p2+3])→p3
160: etf(D$[p2+4,p2+7])→p4
161: itf(D$[p2+8,p2+9])→p5;p5/10→p5
162: itf(D$[p2+10,p2+11])→p6;p6/10→p6
163: itf(D$[p2+12,p2+13])→p7;p7/10→p7;p5+p6+p7→p12
164: itf(D$[p2+30,p2+31])→p8
165: itf(D$[p2+32,p2+33])→p9
166: itf(D$[p2+34,p2+35])→p10
167: itf(D$[p2+36,p2+37])→p11
168: fmt 2x,o47,z
169: wrt r0,"COMP   EXTRA TYPE  DEP   FLY   ONL   FLY   OFF"
170: fmt 3x,o27;wrt r0,"FLY  NEXT  MSN SUB REQ  TOT"
171: fmt 2x,o47,z;wrt r0,"VALUE    COST  A/C ICAO   HRS  ICAO   HRS  ICAO"
172: fmt 3x,o27;wrt r0,"HRS  ICAO  SEQ SEQ NO.  HRS"
173: fmt f7.0,1x,o1,f6.0,2x,f3.0,1x,o4,1x,f5.1,o1,1x,o4,1x,f5.1,z
174: wrt r0,p3,"$",p4,p10,D$[p2+14,p2+17],p5,"→",D$[p2+18,p2+21],p6
175: fmt o1,1x,o4,1x,f5.1,o1,1x,o4,1x,f4.0,1x,f3.0,1x,f3.0,1x,f4.1
*18291
```

```
176: wrt r0, "→", D$[p2+22, p2+25], p7, "→", D$[p2+26, p2+29], p8, p9, p11, p12
177: "dortret": ret
178: "reqtime": 0→p10
179: if p2#130 and p2#141 and p2#5; 400→p5
180: if p2=130; F[1, 4]→p5
181: if p2=141; F[2, 4]→p5
182: if p2=5; F[3, 4]→p5
183: R$[p1+2, p1+5]→L$[1, 4]
184: oll 'ifind'(p6, p7); p10+1→p10
185: itf(I$[p7+4, p7+5])→p8; p8/100→p8
186: itf(I$[p7+6, p7+7])→p9; p9/100→p9
187: oll 'proj'(p8, p9)
188: if p10=1; p8→p11; p9→p12; R$[p1+10, p1+13]→L$[1, 4]; jmp -4
189: (p11-p8)↑2+(p12-p9)↑2→p3; √p3→p3
190: p3/p5→p4; ret
191: "mtime": itf(M$[p1+25, p1+26])→p3
192: itf(M$[p1+33, p1+34])→p4
193: 2400*(p4-p3)→p3; fxd 1
194: itf(M$[p1+31, p1+32])→p4; p3+p4→p4
195: itf(M$[p1+23, p1+24])→p3; p3/100→p3; p4/100→p4
196: 60*int(p4)+100*(p4-int(p4))→p4
197: 60*int(p3)+100*(p3-int(p3))→p3
198: (p4-p3)/60→p2
199: M$[p1+19, p1+22]→L$[1, 4]
200: 1→p10; oll 'ifind'(p3, p4, p10)
*22344
```

```
201: itf(I$[p4+4,p4+5])→p6; p6/100→p6
202: itf(I$[p4+6,p4+7])→p7; p7/100→p7
203: oll 'proj'(p6,p7)
204: M$[p1+27,p1+30]→L$[1,4]
205: 1→p10; oll 'ifind'(p3,p4,p10)
206: itf(I$[p4+4,p4+5])→p8; p8/100→p8
207: itf(I$[p4+6,p4+7])→p9; p9/100→p9
208: oll 'proj'(p8,p9)
209: (p6-p8)↑2+(p7-p9)↑2→p3; fp3→p3; ret
210: "ortline":
211: for K=1 to 79; wtb r0,45; next K; wtb r0,10,13; ret
212: "mcheck": if p1=0 or p2=0; jmp 8
213: rread 2,p1,M$[1,252]
214: itf(M$[p2,p2+1])→p5; p1→p3; p2→p4
215: p4+42→p4
216: if p4>=253; p3+1→p3; 1→p4; rread 2,p3,M$[1,252]
217: itf(M$[p4,p4+1])→p6
218: if p5=p6; jmp -3
219: p4-42→p4; if p4<1; p3-1→p3; 211→p4
220: ret
221: "optort": oll 'rfind'(p11,p1); if r0=10; wtb r0,27,72,27,74
222: rread 4,p11,D$[1,252]; itf(D$[37,38])→p10
223: if p11#p10; prt "REQMT/DIVERSION"; prt "DATA UNMATCHED"; gto "optortret"
224: if p3<p2 or p3>6; p2→p3
225: oll 'rort'(p1)
*27349
```

```
226: for P=p2 to p3
227: oll 'ortline';42(P-1)+1→p13;fmt 29x,o18,f1.0
228: itf(D$[p13+30,p13+31])→p4;wrt r0,"DIVERSION OPTION #",P;fmt 0
229: itf(D$[p13+32,p13+33])→p5
230: 1→p8→p14;oll 'mfind'(p4,p14,p6,p7,p8)
231: oll 'mcheck'(p6,p7,p9,p10)
232: oll 'mort'(p6,p7,p9,p10);oll 'crtline'
233: oll 'dort'(p11,p13);oll 'crtline'
234: next P
235: ret
236: "rfind":if p1>=1 and p1<=9;28(p1-1)+1→p2
237: efg 14;if p1=0 or p1<1 or p1>9 and p2mod28=1;(p2+28-1)/28→p1
238: if p1<1 or p1>9 or p2mod28#1;1→p1→p2
239: ofg 14;ret
240: "space":
*23650
```

APPENDIX G

DESCRIPTION AND HPL CODE FOR MAP PLOT OVERLAY

APPENDIX G

DESCRIPTION AND HPL CODE FOR MAP PLOT OVERLAY

The first section of this Appendix describes the function and HPL calling syntax for each of the modules included in the MAP PLOT overlay. The second section is a listing of the HPL code for these modules.

MODULE FUNCTIONS AND CALLING SYNTAX

Cll 'mapsetup' (flag to prevent prompt messages)

mapsetup - Issues prompt messages to get user map boundary selections, pen colors to be used for plotting, and to select a grid or border to be plotted on the map. Computes map projection parameters based on the map boundaries and the size and aspect ratio of the plotting surface, and assigns these parameters to calculator global variables for use by other modules. Calls subroutines "pens", "grid" and "border" to perform those functions if selected by the user. If "mapsetup" is called with parameter p1 equal to 1, no prompt messages are issued, and the grid, pen selection, border options are not available. Default values are used for map boundaries, and the map projection parameters are computed based on these values. Although it is transparent to the user, the "display" module invokes "mapsetup" in this fashion so there will be usable map projection parameters available for use by other modules.

cll 'pens' (grid pen #, map pen #, missions pen #, reqmt pen #, airfield pen #, diversions pen #)

pens - The six parameters passed to the module, p1 through p6, are taken as pen number selections for plotting the following types of data, respectively: map grid, map coordinates, missions, requirements, airfields, and diversions. These values are printed, and then prompt messages are issued to allow the user to change the pen number designations. Pen number designations are stored with the global variables (see Table I) for use by other modules; this module was designed for those global variables to be used as the calling parameters, since any changes to the pen color designations will automatically change the corresponding global variables.

cll 'grid' (grid size in degrees)

grid - The calling parameter to this module (p1) establishes the size of the map grid to be plotted. Using the current map boundaries from the global variables, the module computes coordinates to construct the grid, calls the subroutine "proj" to convert the coordinates to plotter units, and issues plotter control commands to plot the grid. A legend is printed at the bottom of the plot to identify the size of the grid squares.

cll 'border'

border - Using the plotting surface limits stored with the global variables, this module issues plotter control commands to plot a frame around the plotting surface.

cll 'cdis' (plot flag, print flag)

cdis - This module issues plotter control command to select the pen color designated for plotting map coordinates, and then calls the subroutine "mapplt" to display the map coordinates. Parameters p1 and p2 passed to the module are flags to indicate whether a plot and/or print of the data is selected. A 1 indicates the plot or print is required; a Ø indicates no plot or print.

cll 'mapplt' (plot flag, print flag)

mapplt - Issues prompt messages to get user selections for map coordinate records to be displayed, reads the records selected from the "map" file, and plots and/or prints the coordinates. Parameter p1 is a flag to indicate plotting is desired; p2 indicates a print is desired. Mapplt calls "center" and "arcplt" to plot curved lines between two coordinates.

cll 'center' (X1, Y1, Radius, X2, Y2, XC, YC)

center - Computes the plotter coordinates for the center of a circle which will pass through points designated by p1, p2 and p4, p5, and with a radius of p3. The coordinates for the circle are passed back to the calling program as p6, p7. If p3 is positive, the center will correspond to a counterclockwise arc from p1, p2 to p4, p5; if p3 is negative, the center will correspond to a clockwise arc, and values for p1, p2 and p4, p5 are interchanged.

cll 'arcplt' (X1, Y1, Radius, X2, Y2, XC, YC)

arcplt - Plots a counterclockwise arc with radius p3 from points p1, p2 to p4, p5. If p3 is positive, the plotter remains at p4, p5 when the plot is complete; if p3 is negative, the plotter returns to p1, p2, and then interchanges the values for p1, p2 and p4, p5. The interchange of values done by "center" allows arcplt to produce a clockwise plot from the original point p1, p2 to p4, p5; to leave the plotter on the original p4, p5; and to return the coordinates to their original value.

cll 'angdeg' (X1, Y1, X2, Y2, Angle)

angdeg - Computes the angle between the origin of the plotting surface and two pairs of coordinate values.

```
0: "file1-onefil";
1: "mapsetup";polr;fmt 0;if p1=1;dep "CONTINUE WHEN PLOTTER READY";etp
2: 0→r29→r30;wrt 705,"VS10"
3: pso 705;wrt 705,"IM223,16,0"
4: wrt 705,"OP";red 705,r17,r18,r19,r20
5: -67→r9;-125→r10;38→r12;if p1=0;jmp 4
6: ent "Map East Boundry",r9
7: ent "Map West Boundry",r10
8: ent "Map North-South Midpoint",r12
9: r9+(r10-r9)/2→r11
10: 3438tan(90-r12)→r13
11: sin(r12)→r16
12: r13*sin(r16*(r9-r11))→r14
13: r14(r20-r18)/(r19-r17)→r15
14: sol -r14,r14,-r15,r15
15: lim -r14,r14,-r15,r15;0→r29
16: 3→r22→r23→r27;4→r25;1→r26;2→r28;if p1=0;jmp 6
17: ent "NEW PEN COLORS>>1=YES,0=NO",p2
18: if p2=1;oll 'pens'(r22,r23,r25,r26,r27,r28)
19: 0→p3;ent "Map Grid Sizes No Grid...CONTINUE",p3
20: if p3#0;pen# r22;oll 'grid'(p3)
21: ent "MAP BORDER? 1=YES",p5;if p5=1;oll 'border'
22: ret
23: "pens";
24: prt "CURRENT PEN SET"
25: fmt f1.0
*18015
```

```
26: wrt 16,"Map/Cordes Pen#",p2
27: wrt 16,"Menes Pen#",p3
28: wrt 16,"Reqmtes Pen#",p4
29: wrt 16,"ICAOs Pen#",p5
30: wrt 16,"Diversions Pen#",p6
31: wrt 16,"Grids Pen#",p1
32: if p7=1;gto "penoret"
33: ent "Map/Coorde Pen#",p2;if p2<1 or p2>4;jmp 0
34: ent "Missions Pen#",p3;if p3<1 or p3>4;jmp 0
35: ent "Reqmts Pen#",p4;if p4<1 or p4>4;jmp 0
36: ent "ICAO Pen#",p5;if p5<1 or p5>4;jmp 0
37: ent "Diversions Pen#",p6;if p6<1 or p6>4;jmp 0
38: ent "Grid Pen#",p1;if p1<1 or p1>4;jmp 0
39: 1→p7;gto "pene"
40: "penoret":ret
41: "mapintplt":oll 'mapsetup'(1)
42: pen# r23;0→Y→Z;1→r24
43: asgn "map",1
44: on end 1,"endmapint"
45: oni 7,"oreopltint"
46: eir 7,128
47: ret
48: "endmapint":
49: eir 7,0
50: 0→r24→Y→Z;iret
*18966
```

```
51: "orecpltint":
52: if Z=0; rread 1, Y+1→Y; eread 1, C$[1, 252]
53: itf(C$[Z+1→Z, Z+1→Z])→r31; r31/100→r31→r1
54: itf(C$[Z+1→Z, Z+1→Z])→r32; r32/100→r32→r2
55: itf(C$[Z+1→Z, Z+1→Z])→r33
56: if r33=8256; gto "endmapint"
57: if r33=0; jmp 6
58: oll 'proj'(r31, r32)
59: if r33=1 or r33=2; plt r31, r32, r33; r31→r3; r32→r4; jmp 4
60: oll 'center'(r3, r4, r31, r32, r33, r34, r35)
61: oll 'oroplt'(r3, r4, r31, r32, r33, r34, r35)
62: r31→r3; r32→r4
63: if Z>=252; 0→Z
64: eir 7, 128; iret
65: "oroplt": efg 14; p1→p10; p2→p11
66: deg; oll 'angdeg'(p1, p2, p6, p7, p8)
67: oll 'angdeg'(p3, p4, p6, p7, p9)
68: if p8>=p9; p9+360→p9
69: plt p1, p2, 1
70: "lop": p8+5→p8
71: if p8>=p9; gto "last"
72: abs(p5)cos(p8)+p6→p1
73: abs(p5)sin(p8)+p7→p2
74: plt p1, p2, 2
75: gto "lop"
*1038
```

```
76: "last": plt p3→p8, p4→p9, 2
77: if p5<0; plt p10→p3, p11→p4, 1; p8→p1; p9→p2
78: pen; cfg 14; ret
79: "angdeg":
80: atn((p2-p4)/(p1-p3))→p5
81: if p1<p3; p5+180→p5
82: if p5<0; p5+360→p5
83: ret
    "center": cfg 14
    if p5<0; p1→p8; p2→p9; p3→p1; p4→p2; p8→p3; p9→p4
86: if p2=p4; oll 'equ' (p2, p1, p3, abs(p5), p7, p6); ret
87: if p1=p3; oll 'equ' (p1, p2, p4, abs(p5), p6, p7); ret
88: (p4-p2)/(p3-p1)→p6
89: p4-p6*p3→p7
90: p1+(p3-p1)/2→p8
91: p6*p8+p7→p9
92: p9+p8/p6→p8
93: abs(p5)↑2-p1↑2-p2↑2+2*p2*p8-p8↑2→p9
94: p6*p2-p6*p8-p1p6↑2→p10'
95: p6↑2(1+p6↑2)p9+p10↑2→p9
96: √(p9/(1+p6↑2)↑2)→p9
97: if p6>0 and p2<p4; 0-p9→p9
98: if p6<0 and p2<p4 and p5<0; 0-p9→p9
99: if p6<0 and p2<p4 and p5>0; 0-p9→p9
100: p9-p10/(1+p6↑2)→p9
*28157
```

```
101: p8-p9/p6→p7
102: p9→p6; ofg 14
103: ret
104: "equ": (p3-p2)/2→p6
105: p4↑2-p6↑2→p5
106: if p2<p3; p1-√p5→p5; p2+p6→p6
107: if p2>p3; p1+√p5→p5; p2+p6→p6
108: ret
109: "grid": abs(r10)→p2; p2-p2modp1→p2
110: p2*sgn(r10)→p2; p2-p1→r5; line 1,.5
111: for I=0 to 70 by p1
112: I→r2→r4; r5→r1→r3
113: oll 'proj'(r3,r4); plt r3,r4,1
114: r1+p1→r1→r3; r2→r4
115: oll 'proj'(r3,r4); plt r3,r4,2
116: if r1>=r9; jmp 2
117: jmp -3
118: next I
119: r12-(r9-r10)/2→p3; r12+(r9-r10)/2→p4
120: r5+p1→r5→r3; p3→r4
121: oll 'proj'(r3,r4); plt r3,r4,1
122: r5→r3; p4→r4
123: oll 'proj'(r3,r4); plt r3,r4,2
124: if r5<=r9; jmp -4
125: plt 0,-r15,1; osiz 1,2; opit -12,1; fxd 0
*29863
```

```
126: lbl p1," DEGREE GRID SQUARES"
127: line 1,4; line ; ret
128: "space":
129: "space":
130: "cdie": 0→Y→Z→r24; on end 1,"endmap"; pen# r23
131: dsp "PLOTTER WORKING"; oll 'mapplt'(p1,p2)
132: "endmap": 0→Y→Z; fmt 0; ret
133: "mapplt": fmt o5,f7.2,o5,f6.2,1x,o12,f5.0,o6,f3.0,o7,f3.0; 50→p4
134: 0→p3; ent "SHOW COORDINATES ON CRT? YES=1",p3
135: ent "STARTING REC#>>CONT FOR FULL MAP",Y
136: if Y<1; 1→Y; if Y>50; 50→Y
137: 50→p4; ent "ENDING REC#>>CONT FOR FULL MAP",p4; if p4>50; 50→p4
138: if Z=0; rread 1,Y,C$[1,252]
139: itf(C$[Z+1→Z,Z+1→Z])→r31; r31/100→r31→r1
140: itf(C$[Z+1→Z,Z+1→Z])→r32; r32/100→r32→r2
141: itf(C$[Z+1→Z,Z+1→Z])→r33
142: if r33=8256; gto "endmap"
143: if r33=0; jmp 6
144: oll 'proj'(r31,r32); if p1=0; jmp 5
145: if r33=1 or r33=2; plt r31,r32,r33; r31→r3; r32→r4; jmp 4
146: oll 'center'(r3,r4,r31,r32,r33,r34,r35)
147: oll 'arcplt'(r3,r4,r31,r32,r33,r34,r35)
148: r31→r3; r32→r4
149: if p3=0; jmp 2
150: wrt r0,"LONG=",r1," LAT=",r2,"RADIUS/CODE=",r33," REC#=",Y," INDEX=",Z-5
*18092
```

```
151: if p2=1 and r0#701: 701→r0: jmp -1
152: 10→r0: if Z>=252: 0→Z: Y+1→Y
153: if Y<=p4: jmp -15
154: fmt 0: ret
155: "space":
156: "space":
157: "border": pen# r23: plt -r14,-r15,1: plt -r14,r15,2
158: plt r14,r15,2: plt r14,-r15,2: plt -r14,-r15,2: pen: ret
*17771
```

APPENDIX H

DESCRIPTION AND HPL CODE FOR THE ENTER OVERLAY

APPENDIX H

DESCRIPTION AND HPL CODE FOR THE ENTER OVERLAY

The first section of this Appendix describes the function and HPL calling syntax for each of the modules included in the ENTER overlay. The second section is a listing of the HPL codes for the modules.

MODULE FUNCTIONS AND CALLING SYNTAX

The functions of the following groups of modules are the same, except that each deals with a different data source.

cll 'cenmsn'

cenmsn, cenreq, cenica, cencor, censpc

These modules issue a prompt message, and accept the user's entry selecting whether data should be added, deleted, or changed in the corresponding data base. The module then calls specific subordinate modules to accomplish the desired data modification. No parameters are passed to or from these modules. The module "censpc" changes a disk data file selection, then calls the same modules invokved by cenmsn.

cll 'msnget' (msn record nbr, index)

mget, reqget, iget, cget

These modules issue prompt messages, and accept user inputs to get the data for a data record. "mget" gets all data necessary for mission records, "reqget" gets requirement data, "iget" gets airfield information, and "cget" gets map coordinates data. In each case, the parameters passed to the modules are the data record

and index where the data is to be stored. "reqget" is passed an index only, since there is only one record in the "req" file. When the user has responded to all of the prompt messages with data entries, the data is displayed on the CRT, and the user may elect to correct the data, store it, or delete it.

cll 'addmsn'

addmsn, addreq, addica, addcor

These modules locate the proper record number and index where new data should be stored for each file, then call the appropriate "get" modules to get and store the data to be added.

cll 'delmsn'

delmsn, delreq, delica, delcor

These modules issue a prompt message to have the user identify the particular data to be deleted, find the data and display it on the CRT, and, if the user confirms that the proper data has been found, deletes the data from the file.

cll 'chgmsn'

chgmsn, chgreq, chgica, chgcor

These modules issue a prompt message to have the user identify the data to be changed, find the data and display it on the CRT, and then call the appropriate "get" module to get and store the changed data.

```
0: "twofil-file2":
1: "reqget": if p1>0 and p1<=9; jmp 2
2: ent "REQ NUMBER", p1; if p1<1 or p1>9; jmp 0
3: fti (p1)→L$[1,2]; oll 'rfind'(p1,p2); oll 'rort'(p2)
4: R$[p2+2,p2+27]→L$[3,28]
5: "reqdata": ent "ONLOAD ICAO", L$[3,6]
6: itf(L$[7,8])→p3
7: ent "ONLOAD NOT-EARLIER-THAN TIME", p3; fti (p3)→L$[7,8]
8: itf(L$[9,10])→p4
9: ent "ONLOAD NET DATE (JULIAN)", p4; fti (p4)→L$[9,10]
10: ent "OFFLOAD ICAO", L$[11,14]
11: for I=1 to 4
12: cap(L$[I+2,I+2])→L$[I+2,I+2]
13: cap(L$[I+10,I+10])→L$[I+10,I+10]
14: next I
15: itf(L$[15,16])→p5
16: ent "OFFLOAD NOT-LATER-THAN TIME", p5; fti (p5)→L$[15,16]
17: itf(L$[17,18])→p6
18: ent "OFFLOAD NLT DATE (JULIAN)", p6; fti (p6)→L$[17,18]
19: itf(L$[19,20])→p7
20: ent "CARGO (TONS)", p7; fti (p7)→L$[19,20]
21: itf(L$[21,22])→p8
22: ent "PAX", p8; fti (p8)→L$[21,22]
23: itf(L$[23,24])→p9
24: ent "CARGO SUITABILITY C5/C141/C130", p9; fti (p9)→L$[23,24]
25: fmt 11x,c50
*7306
```

```
26: wrt r0,"SEQ# ONLOAD  NET DAY OFFLOAD  NLT DAY CGO PAX CODE"
27: fmt 13x,f2.0,3x,o4,1x,f4.0,1x,f3.0,4x,o4,1x,f4.0,1x,f3.0,1x,f3.0,z
28: wrt r0,p1,L$[3,6],p3,p4,L$[11,14],p5,p6,p7
29: fmt 1x,f3.0,2x,f3.0;wrt r0,p8,p9
30: ent "CORRECT/STORE=1,REENTER DATA=2",p10
31: if p10=2;gto "reqdata"
32: fti (p1)→L$[1,2];if p10#1;jmp 4
33: L$[1,24]→R$[p2,p2+23];" "→R$[p2+24,p2+26];"0"→R$[p2+27,p2+27]
34: rread 3,1;sprt 3,R$[1,252]
35: oll 'rort'(p2)
36: ret
37: "cenepc":aegn "epeo",2;oll 'cenmen';jmp 6
38: ent "MIeeions, COordinates or QUit",L$[1,2]
39: if L$[1,2]="MI";aegn "epeo",2;oll 'cenmen'
40: if L$[1,2]="CO";aegn "epeo",1;oll 'cenoor'
41: if L$[1,2]="QU";" "→L$[1,2];jmp 2
42: " "→L$[1,2];jmp -4
43: ret
44: "cenmen":
45: ent "ADd, DElete, CHange or QUit",L$[1,2]
46: if L$[1,2]="AD";oll 'addmen'
47: if L$[1,2]="DE";oll 'delmen'
48: if L$[1,2]="CH";oll 'chgmen'
49: if L$[1,2]="QU";" "→L$[1,2];jmp 2
50: " "→L$[1,2];jmp -5
*23781
```

```
51: ret
52: "addmen": 0→p1→p2→p3
53: on end 2,"menpnt"
54: rread 2,p1+1→p1,M$[1,252]
55: " "→L$[1,41]; "@"→L$[42,42]
56: pos(M$,L$[1,42])→p2
57: if p2=0; jmp -3
58: if p2=1 or p2=43 or p2=85 or p2=127 or p2=169 or p2=211; 1→p3
59: if p3=1; oll 'menget'(p1,p2)
60: if p3=0; jmp -6
61: ret
62: "menpnt":
63: p1-1→p1; wrt 16,p1," Records Checked"
64: wrt 16,"No Space for Added Mission"
65: jmp -4
66: "delmen": ent "Sequence No of Man to be Deleted",p1
67: ent "SubSequence No.",p2
68: 1→p5; oll 'mfind'(p1,p2,p3,p4,p5)
69: oll 'mort'(p3,p4)
70: 0→p5; ent "Delete=1, Try Again=2, Done=0",p5
71: if p5=2; jmp -5
72: if p5=1; " "→M$[p4,p4+40]; "@"→M$[p4+41,p4+41]
73: if p5=1; rread 2,p3; eprt 2,M$[1,252]
74: ret
75: "chgmen":
*26776
```

```
76: ent "Sequence No of Man to be Changed",p1
77: ent "Subsequence No.",p2
78: 1→p5;oll 'mfind'(p1,p2,p3,p4,p5)
79: oll 'mort'(p3,p4)
80: 0→p5;ent "Right Man=1, Try Again=2, Done=0",p5
81: if p5=2;jmp -5
82: if p5=1;oll 'menget'(p3,p4)
83: ret
84: "menget":
85: if p2>=42;oll 'mort'(p1,p2-42)
86: oll 'mort'(p1,p2);M$[p2,p2+41]→L$[1,42]
87: "mdata":itf(L$[1,2])→p3
88: ent "Man Sequence No",p3;fti (p3)→L$[1,2]
89: itf(L$[3,4])→p4
90: ent "Man SubSequence No",p4;fti (p4)→L$[3,4];"*"→L$[5,5]
91: ent "Man ID",L$[6,17]
92: itf(L$[18,19])→p5
93: ent "Type A/C: 005,141 or 130",p5;fti (p5)→L$[18,19]
94: ent "Departure ICAO",L$[20,23]
95: itf(L$[24,25])→p6
96: ent "ETD",p6;fti (p6)→L$[24,25]
97: itf(L$[26,27])→p7
98: ent "Departure Julian Day",p7;fti (p7)→L$[26,27]
99: ent "Arrival ICAO",L$[28,31]
100: for I=1 to 4
*21346
```

```
101: cap(L$[19+I,19+I])→L$[19+I,19+I]
102: cap(L$[27+I,27+I])→L$[27+I,27+I]
103: next I
104: itf(L$[32,33])→p8
105: ent "ETA",p8;fti (p8)→L$[32,33]
106: itf(L$[34,35])→p9
107: ent "Arrival Julian Day",p9;fti (p9)→L$[34,35]
108: itf(L$[36,37])→p10
109: ent "Cargo (Tons) on Board",p10;fti (p10)→L$[36,37]
110: itf(L$[38,39])→p11
111: ent "Pax on Board",p11;fti (p11)→L$[38,39]
112: fmt o56,z
113: wrt 10,"SEQ   NO  MISSION ID  TYPE DEPT  DAY  ETD  ARR  DAY  ETA"
114: fmt o10;wrt 10,"  CGO  PAX"
115: fmt 2f4.0,1x,o12,1x,f4.0,1x,o4,1x,f4.0,1x,f4.0,1x,o4,z
116: wrt 10,p3,p4,L$[6,17],p5,L$[20,23],p7,p6,L$[28,31]
117: fmt x,4f4.0;wrt 10,p9," ",p8," ",p10," ",p11
118: ent "Data Correct/store=1;Reenter=0",p12
119: if p12=0;gto "mdata"
120: L$[1,39]→M$[p2,p2+38];"000"→M$[p2+39,p2+41]
121: rread 2,p1;eprt 2,M$[1,252]
122: ret
123: "oenioa":ent "ADd,DElete,CHange or QUit",L$[1,2]
124: if L$[1,2]="AD";oll 'addioa'
125: if L$[1,2]="CH";oll 'chgioa'
*7964
```

```
126: if L$[1,2]="DE";oll 'delica'
127: if L$[1,2]="QU";" "→L$[1,2];jmp 2
128: " "→L$[1,2];jmp -5
129: ret
130: "addica";0→p1→p2→p3
131: on end 5,"noepo"
132: " "→L$[1,9];"0"→L$[10,10]
133: rread 5,p1+1→p1,I$[1,252]
134: poe(I$,L$[1,10])→p2
135: if p2=0;jmp -2
136: if p2mod10#1;jmp -3
137: oll 'iget'(p1,p2)
138: ret
139: "noepo":fmt f2.0
140: wrt 16,p1," RECORDS"
141: wrt 16,"CHECKED-NO SPACE"
142: wrt 16,"IN ICAO FILE";jmp -4
143: "delica":ent "ICAO TO BE DELETED",L$[1,4]
144: oll 'ifind'(p1,p2)
145: oll 'iort'(p1,p2)
146: ent "DELETE=1,TRY AGAIN=2,DONE/EXIT=0",p3
147: if p3=2;jmp -4
148: if p3=1;" "→I$[p2,p2+8];"0"→I$[p2+9,p2+9]
149: if p3=1;rread 5,p1;eprt 5,I$[1,252]
150: ret
*31626
```

```
151: "ohgioa": ent "ICAO TO BE CHANGED", L$[1,4]
152: oll 'ifind'(p1,p2)
153: oll 'iort'(p1,p2)
154: ent "CHANGE=1, TRY AGAIN=2, DONE/EXIT=0", p3
155: if p3=2; jmp -4
156: if p3=1; oll 'iget'(p1,p2)
157: ret
158: "iget": rread 5,p1,I$[1,252]
159: itf(I$[p2+4,p2+5])→p3; p3/100→p3
160: I$[p2,p2+3]→L$[1,4]; ent "ICAO Identifier", L$[1,4]
161: ent "LONGITUDE", p3; fti (p3*100)→L$[5,6]
162: itf(I$[p2+6,p2+7])→p4; p4/100→p4
163: ent "LATITUDE", p4; fti (p4*100)→L$[7,8]
164: itf(I$[p2+8,p2+9])→p5; ent "SUITABILITY CODES C5/C141/C130", p5
165: fti (p5)→L$[9,10]; wrt r0, "ICAO    LONG     LAT  CODES"
166: fmt o4, 1x, f7.2, 1x, f7.2, 2x, f4.0
167: wrt r0, L$[1,4], p3, p4, p5
168: ent "CORRECT/STORE=1, REENTER DATA=0", p6
169: if p6=1; L$[1,10]→I$[p2,p2+9]; rread 5,p1; sprt 5, I$[1,252]
170: if p6=0; jmp -11
171: ret
172: "oenoor":
173: ent "ADd, DElete, CHange, or QUit", L$[1,2]
174: if L$[1,2]="AD"; oll 'addoor'
175: if L$[1,2]="DE"; oll 'deloor'
*8020
```

```
176: if L$[1,2]="CH"; oll 'ohgoor'
177: if L$[1,2]="QU"; " "→L$[1,2]; jmp 2
178: " "→L$[1,2]; jmp -5
179: ret
180: "addoor"; on end 1,"endoor"
181: 0→p2→p3; 7→p1
182: " "→L$[1,5]; "0"→L$[6,6]
183: rread 1,p1+1→p1, C$[1,252]
184: pos(C$, L$[1,6])→p2
185: if p2=0; jmp -2
186: if p2mod6#1; jmp 2
187: oll 'oget'(p1,p2)
188: ret
189: "endoor"; p1-1→p1; fmt 0
190: wrt 16,p1," RECORDS CHECKED"
191: wrt 16,"NO SPACE IN FILE"; jmp -3
192: "deloor"; ent "REC# OF COORDINATE TO BE DELETED",p1
193: if p1<1 or p1>50; prt "INVALID REC#"; jmp -1
194: ent "INDEX OF COORDINATE FOR DELETION",p2
195: if p2<1 or p2>247 or p2mod6#1; prt "INVALID INDEX"; jmp -1
196: oll 'oort'(p1,p2)
197: ent "RIGHT ONE=1, TRY AGAIN=2, QUIT=0",p3
198: if p3=2; jmp -6
199: if p3=1; " "→C$[p2,p2+3]; fti (0)→C$[p2+4,p2+5]
200: if p3=1; rread 1,p1; eprt 1,C$[1,252]
*20354
```

```
201: ret
202: "ohgoor": ent "REC# of COORDINATE TO BE CHANGED", p1
203: if p1<1 or p1>50; prt "INVALID REC#"; jmp -1
204: ent "INDEX OF COORDINATE FOR CHANGE", p2
205: if p2<1 or p2>247 or p2mod6#1; prt "INVALID INDEX"; jmp -1
206: if p2>12; oll 'oort'(p1, p2-12)
207: if p2>6; oll 'oort'(p1, p2-6)
208: oll 'oort'(p1, p2)
209: wrt r0, " ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ "
210: if p2<248; oll 'oort'(p1, p2+6)
211: if p2<242; oll 'oort'(p1, p2+12)
212: ent "RIGHT ONE=1, TRY AGAIN=2, QUIT=0", p3
213: if p3=2; jmp -6
214: if p3=1; oll 'oget'(p1, p2)
215: ret
216: "oort": if p2<1 or p1>50; jmp 8
217: if p2<1 or p2>247 or p2mod6#1; jmp 7
218: rread 1, p1, C$[1, 252]
219: itf(C$[p2, p2+1])→p3; p3/100→p3
220: itf(C$[p2+2, p2+3])→p4; p4/100→p4
221: itf(C$[p2+4, p2+5])→p5
222: fmt o5, f8.2, 2x, o4, f7.2, 2x, o12, f5.0, 2x, o5, f3.0, 2x, o6, f4.0
223: wrt r0, "LONG=", p3, "LAT=", p4, "RADIUS/CODE=", p5, "REC#=", p1, "INDEX=", p2
224: fmt 0; ret
225: "oget": rread 1, p1, C$[1, 252]
*12346
```

```
226: itf(C$[p2,p2+1])→p3;p3/100→p3
227: itf(C$[p2+2,p2+3])→p4;p4/100→p4
228: itf(C$[p2+4,p2+5])→p5
229: ent "LONGITUDE",p3;fti (p3*100)→L$[1,2]
230: ent "LATITUDE",p4;fti (p4*100)→L$[3,4]
231: ent "RADIUS/CODE>neg no for clockwise",p5;fti (p5)→L$[5,6]
232: fmt o5,f9.2,2x,o4,f7.2,2x,o12,f5.0,2x,o5,f4.0,2x,o6,f4.0
233: wrt 10,"LONG=",p3,"LAT=",p4,"RADIUS/CODE=",p5,"REC#=",p1,"INDEX=",p2
234: ent "CORRECT/STORE=1,REENTER DATA=0",p6
235: if p6=0;jmp -6
236: if p6=1;L$[1,6]→C$[p2,p2+5];rread 1,p1;eprt 1,C$[1,252]
237: ret
238: "cenreq":ent "ADd,DElete,CHange or Quit",L$[1,2]
239: if L$[1,2]="AD";oll 'addreq'
240: if L$[1,2]="CH";oll 'chgreq'
241: if L$[1,2]="DE";oll 'delreq'
242: if L$[1,2]="QU";" "→L$[1,2];jmp 2
243: " "→L$[1,2];jmp -5
244: ret
245: "addreq":oll 'reqget';ret
246: "chgreq":oll 'reqget';ret
247: "delreq":ent "REQUIREMENT NBR",p1
248: if p1<1 or p1>9;dsp "VALID NBRS ARE 1 TO 9";wait 3000;jmp -1
249: oll 'rfind'(p1,p2);oll 'rort'(p2)
250: ent "DELETE=1,TRY AGAIN=2,QUIT=0",p3
*21100
```

```
251: if p3=2: jmp -4
252: if p3=1: " "→L$[1,27]: "@"→L$[28,28]
253: if p3=1: L$[1,28]→R$[p2,p2+27]: rread 3,1: eprt 3,R$[1,252]
254: ret
*23327
```

APPENDIX I

THE DISPLAY OVERLAY MODULES DESCRIPTION AND CODE

APPENDIX I

THE DISPLAY OVERLAY MODULES DESCRIPTION AND CODE

The first section of this Appendix describes the function and HPL calling syntax for each of the modules in the DISPLAY overlay. The second section lists the HPL code for these modules.

MODULE FUNCTIONS AND CALLING SYNTAX

The following groups of modules have essentially the same function, except that each deals with a different source of data. Modules prefixed with the letter "m" operate on mission data; modules prefixed with the letter "r" operate on requirement data; modules prefixed with the letter "i" operate on airfield data; and modules prefixed with the letter "d" operate on diversion option data.

cll 'mdis' (plot flag, print flag)

mdis, rdis, idis

These modules issue a prompt message to have the user identify the specific mission, requirement or airfield data to be displayed, locate and present the data on the CRT, and call subordinate plot and/or print modules to plot or print the data identified. Parameters p1 and p2 passed to these modules indicate whether a plot and/or print display has been selected.

cll 'mplt' (msn rec nbr, index nbr)

mplt, rplt, dplt

These modules are passed a record number and index for the data to be plotted. Each reads the mission, requirement, airfield or

diversion data into calculator memory. References in mission, requirement and diversion data are located, and airfield plot and label modules are used to plot each airfield, with an appropriate dotted or dashed line between them. Label modules are called to label the mission, requirement or diversion identified along the plotted line.

cll 'mprt' (mission record nbr, index)

mprt, rprt, iprt, dprt

These modules are passed a record number and index for the data to be printed. These modules change the peripheral address maintained as a global variable from the CRT to the printer, call the appropriate "crt" module to perform the data formatting and printing, and then restore the peripheral address variable back to the CRT.

cll 'mlabel' (mission index nbr)

mlabel, rlabel, dlabel

These modules are passed an index number identifying the label data; they use the global variables for plotter current and last position to compute where to position the label. Each module calls "angdis" to compute the angle for the label data. The plotter is positioned at the proper location along the mission, requirement or diversion line, and the label text is printed along the line and parallel to it.

Functions for the following modules must be described uniquely:

iplt - Module locates airfield data having the record number and index passed as p1 and p2, reads it into calculator memory, mathematically projects the coordinates and converts them to plotter units, and commands the plotter to plot on asterisk superimposed over a zero at that location. Calls the module "ilabel" to label the ICAO identifier.

cll 'ilabel' (airfield index)

ilabel

Locates the ICAO identifier letters identified by the index passed as p1, positions the plotter just below the airfields' coordinates, and commands the plotter to print the ICAO identifier letters.

cll 'mdisall' (plot flag, print flag)

mdisall - This module issues prompt messages to have the user identify a mission, checks the mission data file to locate the record number and index for the first and last mission legs for that mission, then calls the modules "mpltall" and/or "mprtall" to plot or print the complete sequence of mission legs.

cll 'mpltall' (1st msn leg record nbr, index, last msn leg record nbr, index)

mpltall - Calls the "mplt" module to plot each mission leg stored from the record number and index passed as p1 and p2 to the record number and index passed as p3 and p4.

cll 'mprtall' (1st msn leg record nbr, index, last msn leg record nbr, index)

mprtall - Calls the "mprt" module to print data on each mission leg

stored from the record number and index passed as p1 and p2 to the record number and index passed as p3 and p4.

cll 'mlegend'

mlegend - Plots a legend on the lower left hand corner of the plotting surface which depicts and labels each of the three line types used for plotting C5, C141 and C130 mission, requirement or diversion data.

```
0: "file3-trefil":
1: "mlabel":0)p5
2: oll 'angdie'(p2,p3)
3: plt r3,r4,1;iplt (r5-r3)/2,(r6-r4)/2,1
4: (r20-r18)/(r19-r17))p4
5: if p3>=0 and p3<=90 or p3>=270 and p3<=360;ceiz 1,2,p4,p3
6: if p3>90 and p3<270;ceiz 1,2,p4,p3+180;1)p5
7: (r20-r18)/100)p4
8: if p5=0;oplt -15,-1;"<")L$[1,1];M$[p1+5,p1+16])L$[2,13]
9: if p5=0;"<")L$[14,14]
10: if p5=1;oplt 0,-1;">")L$[1,1];M$[p1+5,p1+16])L$[2,13]
11: if p5=1;">")L$[14,14]
12: lbl L$[1,14];pen;ceiz .75,1.5
13: plt r3,r4,1
14: ret
15: "mplt":0)p6
16: if p2mod42#1;gto "mpltret"
17: if p1<1 or p1>200;gto "mpltret"
18: rread 2,p1,M$[1,252]
19: M$[p2+19,p2+22])L$[1,4]
20: 1)p5;oll 'ifind'(p3,p4,p5)
21: if p4#0;oll 'iplt'(p3,p4,1);pen# r25
22: itf(M$[p2+17,p2+18]))p5
23: if p5=5;line 6
24: if p5=141;line 4
*5761
```

```
25: if p5=130;line 2
26: M$[p2+27,p2+30]}L$[1,4];p4}p5
27: 1}p7;oll 'ifind'(p3,p4,p7)
28: if p5#0 and p4#0;oll 'iplt'(p3,p4,2);1}p6;pen# r25
29: if p6#1;oll 'iplt'(p3,p4,1);pen# r25
30: if p5#0 and p4#0;oll 'mlabel'(p2)
31: "mpltret";line ;ret
32: "iplt":
33: if p2=0;gto "ipltret"
34: for I=0 to 24
35: if p2=1+10*I;jmp 3
36: next I
37: gto "ipltret"
38: if p1<1 or p1>25;gto "ipltret"
39: rread 5,p1,I$[1,252]
40: itf(I$[p2+4,p2+5])}p4;p4/100}p4}r1
41: itf(I$[p2+6,p2+7])}p5;p5/100}p5}r2
42: oll 'proj'(p4,p5);plt p4,p5,p3
43: r3}r5;r4}-6;p4}r3;p5}r4
44: oll 'ilabel'(p2,r27)
45: "ipltret";ret
46: "ilabel";if p1<1 or p1>241;gto "ilabelret"
47: pen# p2;osiz 1.5;fmt 0
48: wrt 705,"SM*"
49: plt r3,r4,4
50: wrt 705,"SMo";plt r3,r4,2;pen;wrt 705,"SM"
*30618
```

```
51: pen;oplt -1,-1;osiz 1
52: lbl I$[p1,p1+3];pen
53: plt r3,r4,1
54: "ilabelret":ret
55: "angdis":efg 14;deg
56: (r3-r5)^2+(r4-r6)^2}p1
57: \p1}p1
58: if r3#r5;jmp 3
59: if r6<r4;90}p2;jmp 5
60: if r6>=r4;270}p2;jmp 4
61: atn((r4-r6)/(r3-r5))}p2
62: if r5<r3;p2+180}p2
63: if p2<0;p2+360}p2
64: if p2>360;p2-360}p2
65: ret
66: "mdis":if r29=0 and p1=1;oll 'mlegend'
67: ent "MSN SEQ NO to be Displayed",p3
68: ent "MSN SUBSEQUENCE NO",p4
69: oll 'mfind'(p3,p4,p5,p6,1)
70: oll 'mort'(p5,p6)
71: if p5>0 and p5<201 and p6mod42=1;1}p7
72: ent "RIGHT MSN=1,TRY AGAIN=2,QUIT=0",p8
73: if p8=2;jmp -6
74: if p8=0;jmp 3
75: if p7=1 and p1=1;oll 'mplt'(p5,p6)
*15136
```

```
76: if p7=1 and p2=1; oll 'mprt' (p5, p6)
77: ret
78: "mlegend"; 1)r29; fmt 0
79: osiz .75, 1.5; pen# r25; line 6; plt -r14+5, -r15+r15/20, 1
80: iplt r14/4, 0, 2; lbl " C005"; line 4
81: plt -r14+5, -r15+r15/20, 1; iplt 0, r15/24, 1
82: iplt r14/4, 0, 2; lbl " C141"; line 2
83: plt -r14+5, -r15+r15/20, 1; iplt 0, r15/12, 1
84: iplt r14/4, 0, 2; lbl " C130"; plt -r14+5, -r15+r15/20, 1; pen# r23
85: iplt 0, r15/6, 1; osiz 1; lbl "    LINE TYPES"; line
86: ret
87: "mprt"; 701)r0; if p1#0 and p2#0; jmp 3
88: ent "MSN SEQUENCE NO", p5
89: ent "MSN SUBSEQUENCE NO", p6; oll 'mfind' (p5, p6, p1, p2, 1)
90: oll 'mort' (p1, p2)
91: 10)r0; ret
92: "idie":
93: ent "ICAO to be DISPLAYED", L$[1, 4]
94: 1)p5; oll 'ifind' (p3, p4, p5)
95: oll 'iort' (p3, p4)
96: if p1=1; oll 'iplt' (p3, p4, 1)
97: if p2=1; oll 'iprt' (p3, p4)
98: ret
99: "iprt"; if p1#0 and p2#0; jmp 3
100: ent "ICAO to be Printed", L$[1, 4]
*12680
```

```
101: oll 'ifind' (p1, p2)
102: 701}r0; oll 'iort' (p1, p2); 10}r0; ret
103: "mdieall"; if r29=0 and p1=1; oll 'mlegend'
104: ent "MSN SEQ NO to be Displayed", p8
105: oll 'mfind' (p8, 1, p3, p4, 1)
106: oll 'mohook' (p3, p4, p5, p6)
107: oll 'mort' (p3, p4, p5, p6)
108: 0}p8; ent "RIGHT MSN=1, TRY AGAIN=2, QUIT=0", p8
109: if p8=2; jmp -5
110: if p8=0; jmp 4
111: if p3>0 and p5>0 and p4mod42=1 and p6mod42=1; 1}p7
112: if p7=1 and p1=1; oll 'mpltall' (p3, p4, p5, p6)
113: if p7=1 and p2=1; oll 'mprtall' (p3, p4, p5, p6)
114: ret
115: "mpltall"; p1}p5; p2}p6
116: oll 'mplt' (p5, p6)
117: p6+42}p6; if p6>=253; p5+1}p5; 1}p6
118: if p5>p3 or p5=p3 and p6>p4; jmp 2
119: jmp -3
120: ret
121: "mprtall"; 701}r0; oll 'mort' (p1, p2, p3, p4)
122: 10}r0; ret
123: "optdie"; if p4>=1 and p4<=9; p4}p6; jmp 3
124: "optreqtry"; ent "REQUIREMENT NBR", p6
125: if p6<1 or p6>9; dsp "VALID NBRS ARE 1 to 9"; wait 3000; jmp -1
*3377
```

```
126: rread 4,p6,D$[1,252]
127: if p3<1 or p3>6;ent "FIRST OPTION# TO BE PRESENTED",p3
128: if p3<1 or p3>6;jmp -1
129: if p5<1 or p5>6 or p5<p3;ent "LAST OPTION# TO BE PRESENTED",p5
130: if p5<p3 or p5>6;dsp "INVALID ENTRY";wait 3000;jmp -3
131: itf(D$[(p3-1)42+37,(p3-1)42+38])}p4
132: if p4#p6;prt "REQMT/DIVERSION";prt "DATA UNMATCHED";gto "optdieret"
133: oll 'rfind'(p4,p8);oll 'rort'(p8)
134: ent "RIGHT REQMT? 1=YES 2=NO 0=QUIT",p9
135: if p9=0;gto "optdieret"
136: if p9=2;gto "optreqtry"
137: 0}p7;ent "CRT DISPLAY? 1=YES",p7;if p7=1;oll 'optort'(p8,p3,p5)
138: if p1=1 and r30=0;oll 'optlegend'
139: if p1=1;oll 'optplt'(p8,p3,p5)
140: if p2=1;oll 'optprt'(p8,p3,p5)
141: "optdieret";ret
142: "optprt";701}r0
143: fmt 24x,o30;wrt r0,"OPPORTUNE RESCHEDULING OPTIONS"
144: oll 'optort'(p1,p2,p3);10}r0;ret
145: "rdie";ent "REQ# TO BE DISPLAYED",p3
146: if p3<1 or p3>9;dsp "VALID NBRS ARE 1 TO 9";wait 3000;jmp -1
147: oll 'rfind'(p3,p4);oll 'rort'(p4)
148: ent "RIGHT REQMT=1 TRY AGAIN=2 DONE=0",p5
149: if p5=2;jmp -4
150: if p5=0;jmp 3
*32266
```

```
151: if p1=1;oll 'rplt'(p4)
152: if p2=1;oll 'rprt'(p4)
153: ret
154: "rprt";701}r0;oll 'rort'(p1);10}r0;ret
155: "rplt";rread 3,1,R$[1,252]
156: if p1mod28#1;gto "rpltret"
157: R$[p1+2,p1+5]}L$[1,4];oll 'ifind'(p3,p4,1)
158: oll 'iplt'(p3,p4,1);line ;pen# r26
159: R$[p1+10,p1+13]}L$[1,4];oll 'ifind'(p5,p6,1)
160: if p4#0 and p6#0;oll 'iplt'(p5,p6,2);pen# r26;oll 'rlabel'(p1);jmp 2
161: oll 'iplt'(p5,p6,1)
162: ret
163: "rlabel";itf(R$[p1,p1+1])}p6;" "}L$
164: oll 'angdie'(p3,p4)
165: plt r3,r4,1;iplt (r5-r3)/2,(r6-r4)/2,1
166: (r20-r18)/(r19-r17)}p5;fxd 0
167: if p4>=0 and p4<=90 or p4>=270 and p4<=360;oeiz 1,2,p5,p4
168: if p4>90 and p4<270;oeiz 1,2,p5,p4+180;1}p2
169: if p2=0;oplt 0,-1;"<REQ#"}L$[1,5]
170: if p2=1;oplt 0,-1;">REQ#"}L$[1,5]
171: str(p6)}L$[6];len(L$)}p7;" "}L$[p7+1,p7+1]
172: str(p3)}L$[p7+2];len(L$)}p7
173: if p2=0;" NM<"}L$[p7+1,p7+4]
174: if p2=1;" NM>"}L$[p7+1,p7+4]
175: len(L$)}p7;lbl L$[1,p7];pen;oeiz .75,1.5
*21946
```

```
176: plt r3,r4,1;ret
177: "dplt";if p1<1 or p1>9 or p2mod42#1;gto "dpltret"
178: rread 4,p1,D$[1,252]
179: D$[p2+14,p2+17]}L$[1,4];oll 'ifind'(p3,p4,1)
180: oll 'iplt'(p3,p4,1)
181: itf(D$[p2+34,p2+35])}p5
182: if p5=5;line 6
183: if p5=141;line 4
184: if p5=130;line 2
185: D$[p2+18,p2+21]}L$[1,4];oll 'ifind'(p7,p8,1)
186: if p3=p7 and p4=p8;jmp 4
187: if p4#0 and p8#0;pen# r28;oll 'iplt'(p7,p8,2);jmp 2
188: oll 'iplt'(p7,p8,1);jmp 2
189: pen# r28;oll 'dlabel'(p2,1)
190: D$[p2+22,p2+25]}L$[1,4];oll 'ifind'(p3,p4,1)
191: oll 'iplt'(p3,p4,1)
192: D$[p2+26,p2+29]}L$[1,4];oll 'ifind'(p7,p8,1)
193: if p3=p7 and p4=p8;jmp 4
194: if p4#0 and p8#0;pen# r28;oll 'iplt'(p7,p8,2);jmp 2
195: oll 'iplt'(p7,p8,1);jmp 2
196: pen# r28;oll 'dlabel'(p2,2)
197: "dpltret";ret
198: "dlabel";itf(D$[p1+30,p1+31])}p4
199: itf(D$[p1+32,p1+33])}p5
200: oll 'mfind'(p4,p5,p6,p7,1);oll 'mlabel'(p7)
*16959
```

```
201: ret
202: "optplt"; oll 'rfind' (p12, p1); oll 'rplt' (p1); if p2<1 or p2>6; 1}p2
203: (p2-1)42+1}p13; if p3<p2 or p3>6; p2}p3
204: rread 4, p12, D$[1, 252]; itf(D$[p13+36, p13+37])}p11
205: if p12#p11; prt "REQMT/DIVERSION"; prt "DATA UNMATCHED"; gto "optpltret"
206: for S=p2 to p3
207: oll 'dplt' (p11, 42(S-1)+1}p4)
208: itf(D$[p4+30, p4+31])}p5
209: 1}p9}p6; oll 'mfind' (p5, p6, p7, p8, p9)
210: oll 'mcheck' (p7, p8, p10, p14)
211: oll 'mpltall' (p7, p8, p10, p14)
212: next S
213: ret
214: "optlegend"; 1}r30; csiz 1; if r29=0; oll 'mlegend'
215: plt r14-2, -r15+r15/20, 1; oplt -20, 0; pen# r28; lbl "POSSIBLE DIVERSIONS"
216: plt r14-2, -r15+r15/20, 1; oplt -20, 1; pen# r26; lbl "REQUIREMENTS"
217: plt r14-2, -r15+r15/20, 1; oplt -20, 2; pen# r25; lbl "SCHEDULED MISSIONS"
218: pen# r23; plt r14-2, -r15+r15/15, 1; oplt -16, 3; lbl "LINE COLORS"; pen
219: ret
*2045
```

APPENDIX J

THE OPTIMIZE OVERLAY MODULES DESCRIPTION AND CODE

APPENDIX J

THE OPTIMIZE OVERLAY MODULES DESCRIPTION AND CODE

The first section of this Appendix describes the function and HPL calling syntax for each of the modules included in the OPTIMIZE overlay. The second section lists the HPL codes for the modules.

cll 'weight'

weight - Issues prompt messages and accepts user inputs for new optimization weight factors maintained as global variables. Calls "weightprt" to print current values for each of the weight parameters.

cll 'divset' (requirement nbr)

divset - Clears all data from a single record of the "div" file, and sets the record cost parameters so that new data can be put in the file. The "div" record cleared corresponds to the requirement number passed to the module.

cll 'optreq' (selection flag)

optreq - This module controls a number of activity modules to sequentially read and analyze mission data, compute diversion routes and costs, and to store the computed data if it qualifies as one of the six-best diversion options. The parameter passed to the module is a flag which indicates whether to compute the six-best options, the six next-best options, etc.

cll 'cost' (requirement nbr, 1st option cost, 2nd option cost,...6th option cost)

cost - Reads the "div" file diversion options stored for the requirement number passed as p1, and passes back the costs for each of the six options as p2 through p7.

cll 'mincost' (requirement nbr)

mincost - Calls "cost" to get the cost factors for a set of diversion options, updates the max cost global variable, and calls "divset" to clear the diversion option data from the file.

cll 'eligtype' (requirement nbr, suitability parameter)

eligtype - Computes an airfield suitability parameter for the requirements identified by p1, and passes the parameter back to the calling program as p2. The parameter is used by the calling program to ascertain whether a given mission aircraft can operate into the requirement onload and offload airfields.

cll 'eligcargo' (requirement index, mission leg index, cargo eligibility flag)

eligcargo - Compares the cargo suitability code for the requirement identified by p1 against the aircraft type for the mission identified by p2, and passes a Ø back if the cargo is not compatible with the mission.

cll 'onloadtime' (requirement index, mission leg index, route distance, flying time)

onloadtime - Computes distance and flying time for the mission identified by p2 to fly from its departure station to the onload airfield of the requirement identified by p1.

cll 'offloadtime' (requirement index, mission leg index, route distance, flying time)

offloadtime - Computes distance and flying time for the mission identified by p2 to fly from the offload station of the requirement identified by p1 to the mission's originally-scheduled arrival airfield.

cll 'divmaxtime'

divmaxtime - A dummy module which was designed to determine if a mission's aircrew had sufficient crew duty time remaining to divert to the cargo or passenger requirement. Sufficient aircrew data is not yet available to implement this module.

cll 'eligcat' (mission leg index, diversion-eligible flag)

eligcat - This module checks the mission leg identified by p1, and passes a Ø back to the calling program as p2 if it is not a diversion-eligible mission leg.

cll 'eligtime' (requirement index, mission leg index time, eligibility flag)

eligtime - Checks the mission leg identified by p2 against the requirement identified by p1, and passes a Ø back as p3 if the requirement on-load and offload times cannot be met by diverting the mission.

cll 'eligicao' (mission leg index, airfield suitability parameter, airfield eligibility flag)

eligicao - Checks the type aircraft for the mission leg identified by p1 against the airfield suitability parameter for the airlift requirement computed by "eligtype" and passed as p2 and passes a Ø back as p3 if the aircraft cannot operate into the requirement onload or offload airfields.

cll 'divmin' (req index, msn leg index, weighted cost, raw cost, flying time to onload, requirement flying time, flying time to offload)

divmin - Checks the cost of diverting the mission leg identified by p2 against the costs of previously-selected diversion options; if the cost is less than any of the options already stored, the module formats and stores the new diversion option in the "div" file.

```
0: "forfil-file4":
1: "reqget": if p1>0 and p1<=9; jmp 2
2: ent "REQ NUMBER", p1; if p1<1 or p1>9; jmp 0
3: fti (p1)}L$[1,2]; cll 'rfind' (p1,p2); cll 'rort' (p2)
4: R$[p2+2,p2+27]}L$[3,28]
5: "reqdata": ent "ONLOAD ICAO", L$[3,6]
6: itf(L$[7,8])}p3
7: ent "ONLOAD NOT-EARLIER-THAN TIME", p3; fti (p3)}L$[7,8]
8: itf(L$[9,10])}p4
9: ent "ONLOAD NET DATE (JULIAN)", p4; fti (p4)}L$[9,10]
10: ent "OFFLOAD ICAO", L$[11,14]
11: for I=1 to 4
12: cap(L$[I+2,I+2])}L$[I+2,I+2]
13: cap(L$[I+10,I+10])}L$[I+10,I+10]
14: next I
15: itf(L$[15,16])}p5
16: ent "OFFLOAD NOT-LATER-THAN TIME", p5; fti (p5)}L$[15,16]
17: itf(L$[17,18])}p6
18: ent "OFFLOAD NLT DATE (JULIAN)", p6; fti (p6)}L$[17,18]
19: itf(L$[19,20])}p7
20: ent "CARGO (TONS)", p7; fti (p7)}L$[19,20]
21: itf(L$[21,22])}p8
22: ent "PAX", p8; fti (p8)}L$[21,22]
23: itf(L$[23,24])}p9
24: ent "CARGO SUITABILITY C5/C141/C130", p9; fti (p9)}L$[23,24]
*9000
```

```
25: fmt 11x,c50
26: wrt r0,"SEQ# ONLOAD  NET DAY OFFLOAD  NLT DAY CGO PAX CODE"
27: fmt 13x,f2.0,3x,c4,1x,f4.0,1x,f3.0,4x,c4,1x,f4.0,1x,f3.0,1x,f3.0,z
28: wrt r0,p1,L$[3,6],p3,p4,L$[11,14],p5,p6,p7
29: fmt 1x,f3.0,2x,f3.0;wrt r0,p8,p9
30: ent "CORRECT/STORE=1,REENTER DATA=2",p10
31: if p10=2;gto "reqdata"
32: fti (p1)}L$[1,2];if p10#1;jmp 4
33: L$[1,24]}R$[p2,p2+23];" "}R$[p2+24,p2+26];"@"}R$[p2+27,p2+27]
34: rread 3,1;sprt 3,R$[1,252]
35: cll 'rert'(p2)
36: ret
37: "optreq":0}p8}p23;if p1=1;0}r39
38: cll 'rfind'(p29,p2);if p1>1;cll 'mincost'(p2)
39: rread 3,1,R$[1,252];rread 4,p29,D$[1,252]
40: cll 'eligtype'(p2,p30)
41: for P=1 to 200
42: " "}L$[1,41];"@"}L$[42,42];" "}L$[43,83];"@"}L$[84,84];" "}L$[85,124]
43: rread 2,P,M$[1,252]
44: if M$[1,124]=L$[1,124];prt "LAST MSN REC",P;gto "lastrec"
45: for Q=1 to 211 by 42
46: cll 'eligcat'(Q,p8);if p8=0;gto "nextq"
47: cll 'eligtime'(p2,Q,p8);if p8=0;gto "nextq"
48: cll 'eligicao'(Q,p30,p8);if p8=0;gto "nextq"
49: cll 'eligcargo'(p2,Q,p8);if p8=0;gto "nextq"
50: itf(M$[Q+17,Q+18])}p8
*1331
```

```
51: cll 'reqtime' (p2, p8, p9, p10)
52: cll 'mtime' (Q, p12, p11)
53: cll 'onloadtime' (p2, Q, p13, p14); if p14>12; gto "nextq"
54: cll 'offloadtime' (p2, Q, p15, p16); if p16>12; gto "nextq"
55: 2}p24; if p8=130; 1}p24
56: if p8=141; 2}p24
57: if p8=5; 3}p24
58: 0}p19; if p15>4; p16+F[p24, 2]}p16; p19+1}p19
59: if p13>4; p14+F[p24, 2]}p14; p19+1}p19
60: p10+F[p24, 2]}p10; if p19=0; p12}p10
61: p10+p14+p16-p12}p20
62: cll 'divmaxtime' (Q, p19, p20, p23); if p23#1; gto "nextq"
63: F[p24, 1]*p20}p21; F[p24, 3]*p21}p22
64: cll 'divmin' (p2, Q, p22, p21, p14*10, p10*10, p16*10)
65: "nextq": dep "OPTIMIZATION IN PROGRESS"; next Q
66: next P
67: "lastrec": rread 4, p29; sprt 4, D$[1, 252]
68: cll 'optcrt' (p2, 1, 5); ret
69: "divmin": if p3<r39; gto "divminret"
70: 0}p14; cll 'rfind' (p15, p1); cll 'cost' (p1, p8, p9, p10, p11, p12, p13)
71: if p3>p13; gto "divminret"
72: if p3<p8; 6}p14
73: if p3>=p8 and p3<p9; 5}p14
74: if p3>=p9 and p3<p10; 4}p14
75: if p3>=p10 and p3<p11; 3}p14
*396
```

```
76: if p3>=p11 and p3<p12;2}p14
77: if p3>=p12 and p3<p13;1}p14
78: if p14=0;gto "divminret"
79: for I=1 to p14
80: D$[253-42*I,294-42*I]}D$[min(211,295-42*I),min(252,336-42*I)]
81: next I
82: fte (p3)}L$[1,4];fte (p4)}L$[5,8]
83: fti (p5)}L$[9,10];fti (p6)}L$[11,12];fti (p7)}L$[13,14]
84: M$[p2+19,p2+22]}L$[15,18]
85: R$[p1+2,p1+5]}L$[19,22];R$[p1+10,p1+13]}L$[23,26]
86: M$[p2+27,p2+30]}L$[27,30]
87: M$[p2,p2+3]}L$[31,34];M$[p2+17,p2+18]}L$[35,36]
88: R$[p1,p1+1]}L$[37,38];" "}L$[39,41];"@"}L$[42,42]
89: L$[1,42]}D$[253-42*p14,294-42*p14]
90: rread 4,p15;eprt 4,D$[1,252]
91: "divminret":ret
92: "onloadtime":0}p10
93: itf(M$[p2+17,p2+18])}p5
94: if p5#130 and p5#141 and p5#5;410}p5
95: if p5=130;F[1,4]}p5
96: if p5=141;F[2,4]}p5
97: if p5=5;F[3,4]}p5
98: M$[p2+19,p2+22]}L$[1,4]
99: 1}p12;oll 'ifind'(p6,p7,p12);p10+1}p10;if p12=0;99999}p3;jmp 6
100: itf(I$[p7+4,p7+5])}p8;p8/100}p8
*17332
```

```
101: itf(I$[p7+6, p7+7])}p9; p9/100}p9
102: oll 'proj' (p8, p9)
103: if p10=1; p8}p10; p9}p11; R$[p1+2, p1+5]}L$[1, 4]; jmp -4
104: (p10-p8)^2+(p11-p9)^2}p3; \p3}p3
105: p3/p5}p4; ret
106: "offloadtime": 0}p10
107: itf(M$[p2+17, p2+19])}p5
108: if p5#130 and p5#141 and p5#5; 410}p5
109: if p5=130; F[1, 4]}p5
110: if p5=141; F[2, 4]}p5
111: if p5=5; F[3, 4]}p5
112: R$[p1+10, p1+13]}L$[1, 4]
113: 1}p12; oll 'ifind' (p6, p7, p12); p10+1}p10; if p12=0; 99999}p3; jmp 6
114: itf(I$[p7+4, p7+5])}p8; p8/100}p8
115: itf(I$[p7+6, p7+7])}p9; p9/100}p9
116: oll 'proj' (p8, p9)
117: if p10=1; p8}p10; p9}p11; M$[p2+27, p2+30]}L$[1, 4]; jmp -4
118: (p10-p8)^2+(p11-p9)^2}p3; \p3}p3
119: p3/p5}p4; ret
120: "eligicao": 1}p3
121: itf(M$[p1+17, p1+18])}p4
122: if p4=141; 10}p5
123: if p4=130; 1}p5
124: if p4=5; 100}p5
125: if band(p5, p2)=0; 0}p3
*9565
```

```
126: ret
127: "eligcat":1}p2
128: if M$[p1+39,p1+39]#"D";0}p2
129: ret
130: "eligtime":1}p3
131: itf(R$[p1+16,p1+17])}p4;24*p4}p4
132: itf(M$[p2+33,p2+34])}p5;24*p5}p5
133: itf(R$[p1+8,p1+9])}p6;24*p6}p6
134: itf(M$[p2+25,p2+26])}p7;24*p7}p7
135: itf(R$[p1+6,p1+7])}p8;itf(R$[p1+14,p1+15])}p9
136: itf(M$[p2+23,p2+24])}p10;itf(M$[p2+31,p2+32])}p11
137: p7+int(p10/100)}p12;p5+int(p11/100)}p13
138: p4+int(p9/100)}p14;p6+int(p8/100)}p15
139: if p12>p14;0}p3;jmp 3
140: if M$[p2+6,p2+6]="J" and p12<p15-36;0}p3;jmp 2
141: if M$[p2+6,p2+6]="V" and p12<p15-48;0}p3
142: ret
143: "diveet":fte (S999999)}L$[1,4]}L$[5,8]
144: " "}L$[9,41];"@"}L$[42,42]
145: for I=1 to 211 by 42
146: L$[1,42]}D$[I,I+41]
147: next I
148: oll 'rfind'(p2,p1)
149: rread 4,p2;sprt 4,D$[1,252]
150: ret
*2884
```

```
151: "divmaxtime": 1}p4; ret
152: "cost": cll 'rfind' (p8, p1); rread 4, p8, D$[1, 252]
153: etf(D$[1, 4])}p2; etf(D$[43, 46])}p3
154: etf(D$[85, 88])}p4; etf(D$[127, 130])}p5
155: etf(D$[169, 172])}p6; etf(D$[211, 214])}p7
156: ret
157: "mincost": cll 'cost' (p1, p2, p3, p4, p5, p6, p7)
158: max(p2, p3, p4, p5, p6, p7)}r39
159: cll 'divset' (p1)
160: ret
161: "eligcargo": itf(M$[p2+17, p2+18])}p4; 1}p3
162: itf(R$[p1+22, p1+23])}p5
163: if p5=0 or p5=8224; 111}p5
164: if p4=130; 1}p4
165: if p4=141; 10}p4
166: if p4=5; 100}p4
167: if band(p4, p5)=0; 0}p3
168: ret
169: "weight": 1}F[1, 3]}F[2, 3]}F[3, 3]; 0}p2
170: . 4}F[1, 2]}F[2, 2]}F[3, 2]
171: 280}F[1, 4]; 410}F[2, 4]; 430}F[3, 4]
172: 708}F[1, 1]; 1936}F[2, 1]; 6302}F[3, 1]
173: if p1=0; gto "weightret"
174: cll 'weightprt'
175: ent "CHANGE FLY HR COSTS? 1=YES", p2; if p2=0; jmp 4
*28103
```

```
176: ent "C130 COST/HR",F[1,1]
177: ent "C141 COST/HR",F[2,1]
178: ent "C5 COST/HR",F[3,1];0}p2
179: ent "CHANGE EXTRA FLY TIME/STOP 1=YES",p2;if p2=0;jmp 4
180: ent "C130 EXTRA FLY TIME/STOP",F[1,2]
181: ent "C141 EXTRA FLY TIME/STOP",F[2,2]
182: ent "C5 EXTRA FLY TIME/STOP",F[3,2];0}p2
183: ent "CHANGE WEIGHT FACTORS/EACH ACFT",p2;if p2=0;jmp 4
184: ent "C130 WEIGHT FACTOR",F[1,3]
185: ent "C141 WEIGHT FACTOR",F[2,3]
186: ent "C5 WEIGHT FACTOR",F[3,3];0}p2
187: ent "CHANGE ACFT BLOCK SPEEDS? 1=YES",p2;if p2=0;jmp 4
188: ent "C130 BLOCK SPEED",F[1,4]
189: ent "C141 BLOCK SPEED",F[2,4]
190: ent "C5 BLOCK SPEED",F[3,4]
191: oll 'weightprt'
192: "weightret";ret
193: "eligtype";R$[p1+2,p1+5]}L$[1,4]
194: oll 'ifind'(p3,p4,p5)
195: if p5=0;0}p2;gto "eligtyperet"
196: itf(I$[p4+8,p4+9])}p2
197: R$[p1+10,p1+13]}L$[1,4]
198: oll 'ifind'(p3,p4,p5)
199: if p5=0;0}p2;gto "eligtyperet"
200: itf(I$[p4+8,p4+9])}p6
*4528
```

```
201: band(p2,p6)}p2
202: "eligtyperet":ret
203: "weightprt":epc ;prt "     CURRENT"
204: prt "WEIGHT   FACTORS";epc
205: prt "FLYING HR COSTS"
206: fmt c4,2x,"$",f4.0,"/HR"
207: wrt 16,"C130",F[1,1];wrt 16,"C141",F[2,1]
208: wrt 16,"C005",F[3,1]
209: prt "EXTRA FLY TIME";prt "REQ'D PER STOP"
210: fmt c4,2x,f3.1,1x,"HRS"
211: wrt 16,"C130",F[1,2];wrt 16,"C141",F[2,2];wrt 16,"C005",F[3,2]
212: prt "RESCHEDULING";prt "WEIGHT FACTORS"
213: fmt c4,3x,f5.2
214: wrt 16,"C130",F[1,3];wrt 16,"C141",F[2,3];wrt 16,"C005",F[3,3]
215: prt "ACFT BLOCK SPEED";fmt c4,2x,f4.0," KNOTS"
216: wrt 16,"C130",F[1,4];wrt 16,"C141",F[2,4];wrt 16,"C005",F[3,4]
217: ret
*23825
```

APPENDIX K

MAP PROJECTION FORMULAS

APPENDIX K

MAP PROJECTION FORMULAS

Simple Conic Projection

La = Latitude of a point on earth's surface to be projected to a planar representation.

Lo = Longitude of a point on earth's surface to be projected to a planar representation.

Lm = Latitude of selected north-south midpoint of the map projection. The *cone of projection* is tangent to the earth's surface across this Latitude (see Fig K-1).

Lom = Longitude of selected east-west midpoint of the map projection.

Re = Radius of Earth = 3438 NM (see Fig K-1)

Rm = Conic radius of the map projections north-south midpoint (see Fig K-1)

Rm = Re tan/90-Lm) (Ref 20:105) (see Fig K-1)

R = Conic radius of a point on earth's surface to projected. Corresponds to Lo, La in geographic coordinates.

R = Rm + Re tan (Lm - L) (Ref 18:10) (See Fig K-1)

N = Constant of Tangent Cone; Proportion of angle between meridians on projection and the actual change of Longitude.

N = sin (Lm) (Ref 1:2-27) (see Fig K-2)

The developed cone in the plane will be 360^{o}(N). Each degree of Longitude will be N degrees on the developed cone.

If the origin of a plot is at the point Lm, Lom:

x = R sin N (Lo-Lom) (see Fig K-2)

y = Rm - RcosN (lo-Lom) (Ref 18:10)

the requirement identified by p1.

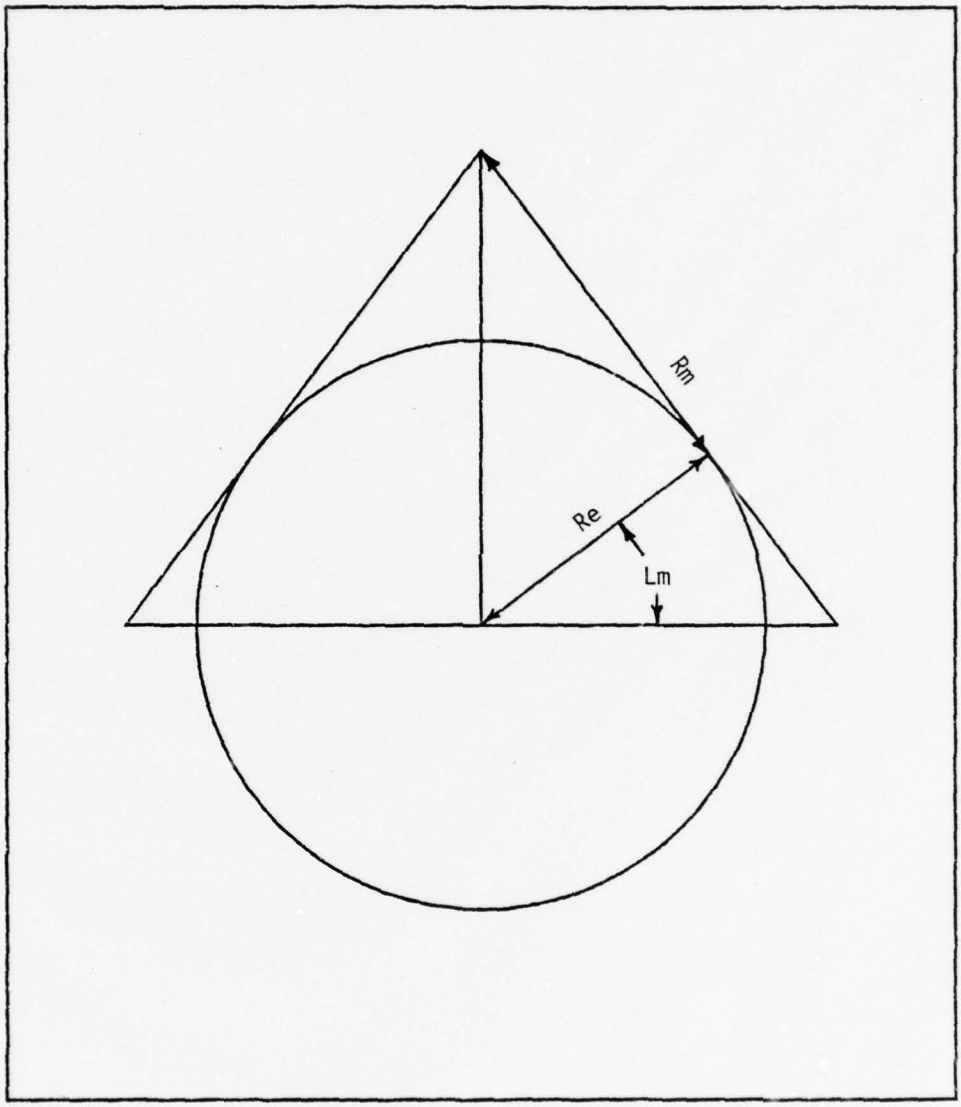


Fig K-1. Cone of Projection Tangent to Earth's Surface

or offload airfields.

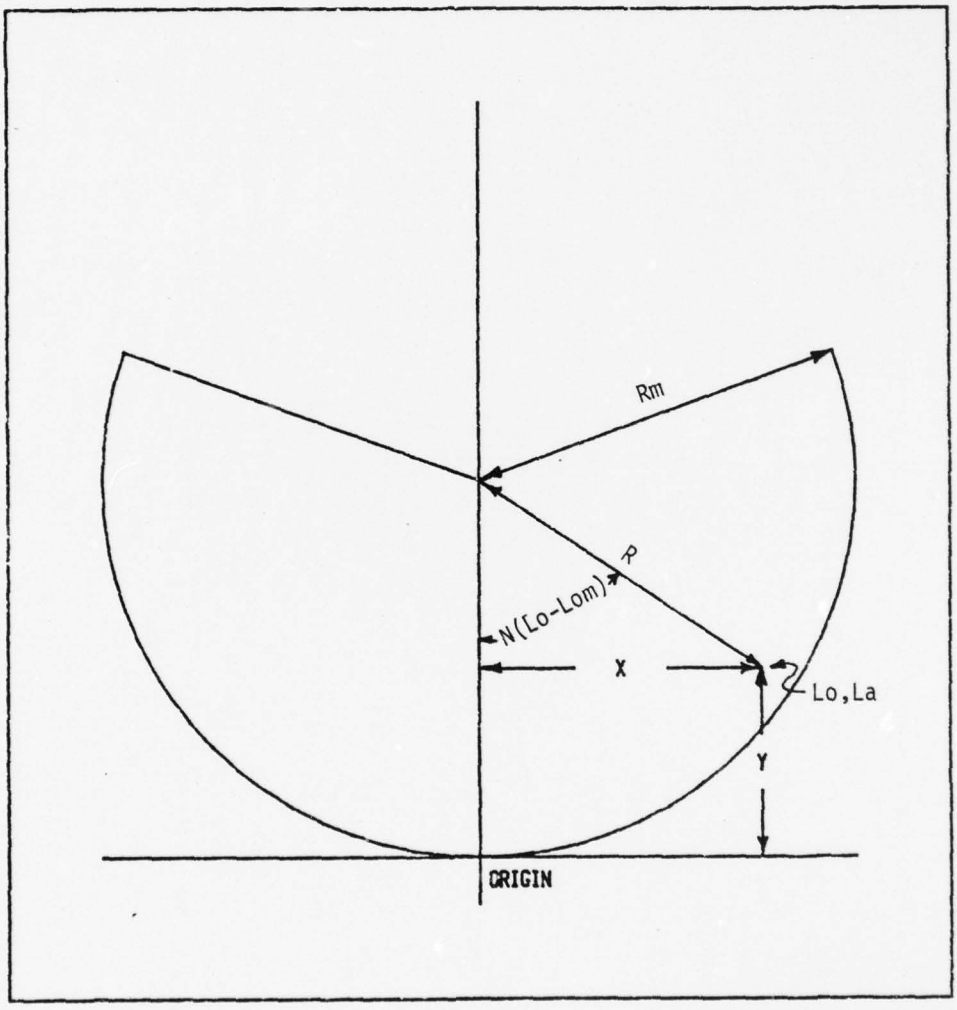


Fig K-2. The Developed Cone in the Plane

APPENDIX L

DATA RECORD FORMATS

APPENDIX L

DATA RECORD FORMATS

All opportune rescheduling system data are stored on HP disk files. Each file consists of a pre-determined number of 256-byte records. Data in the records may be stored in ASCII-coded strings, full-precision numerical representation (8 bytes), split precision numerical (4 bytes), or integer precision numerical (2 bytes). When data is stored as ASCII strings, 4 bytes of each disk record are unusable.

Data record format for the "msns" and "spec" files is shown in Fig N-1; format for the "req" file is shown in Fig N-2; "div" data format in Fig N-3; "map" data format in Fig N-4; and "icao" data in Fig N-5.

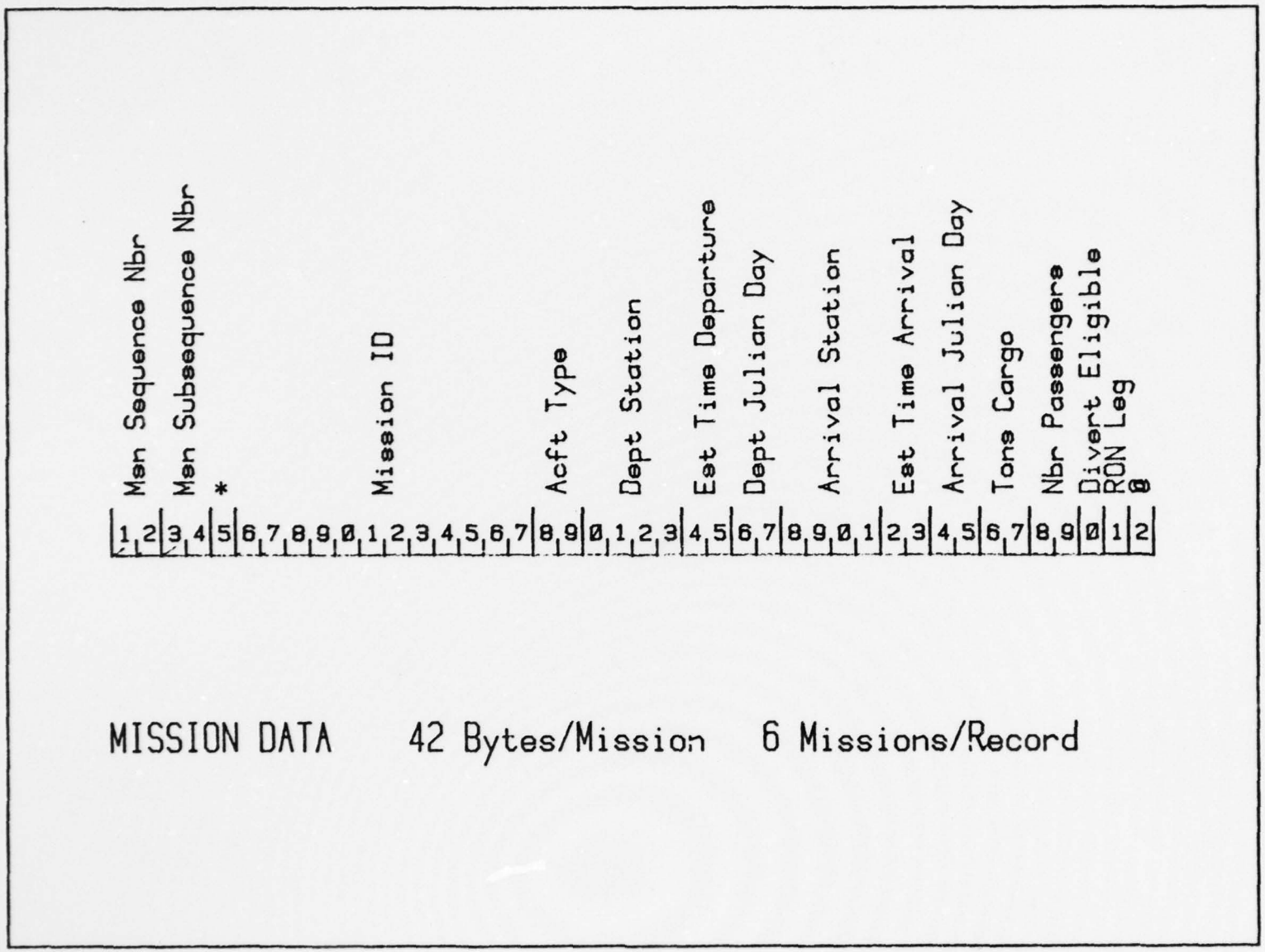


Fig L-1. Data Format for "msns" and "spec" file records

REQUIREMENT DATA 28 Bytes/Reqmt 9 Reqmts/Record

1 2	3 4	5 6	7 8	9 0	1 2	3 4	5 6	7 8	9 0	1 2	3 4	5 6	7 8
Sequence Nbr		Onload ICAO		Not-Earlier-Than Time		NET Julian Day		Offload ICAO		Not-Later-Than Time		NLT Julian Day		Tons Cargo		Nbr Passengers		Acft Compatibility Codes		Filler		"@"	

Fig L-2. Data Format for "req" File Records

DIVERSION DATA 42 Bytes/Diversion 6 Diversions/Record

1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	2

Weighted Cost Factor

Additional Operating Cost

Fly Hrs X10 Dept to Onload

Fly Hrs X10 Onload to Offload

Fly Hrs X10 Offload to Man Arrv

Departure ICAO

Onload ICAO

Offload ICAO

Man Arrival ICAO

Man Sequence Nbr

Man Subsequence Nbr

Type Aircraft

Requirement Nbr

Filler

"0"

Fig L-3. Data Format for "div" File Records

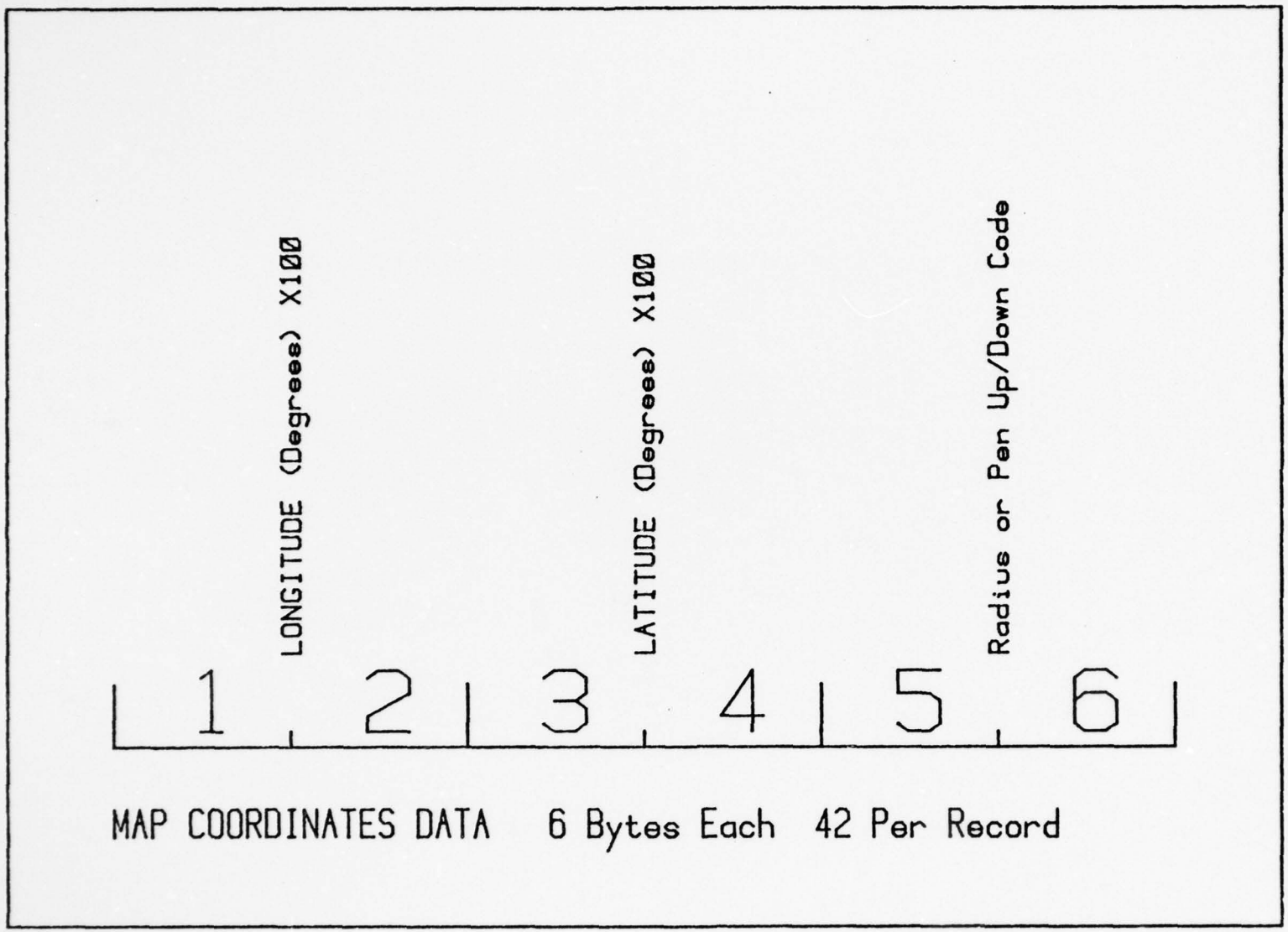


Fig L-4. Data Format for "map" File Records

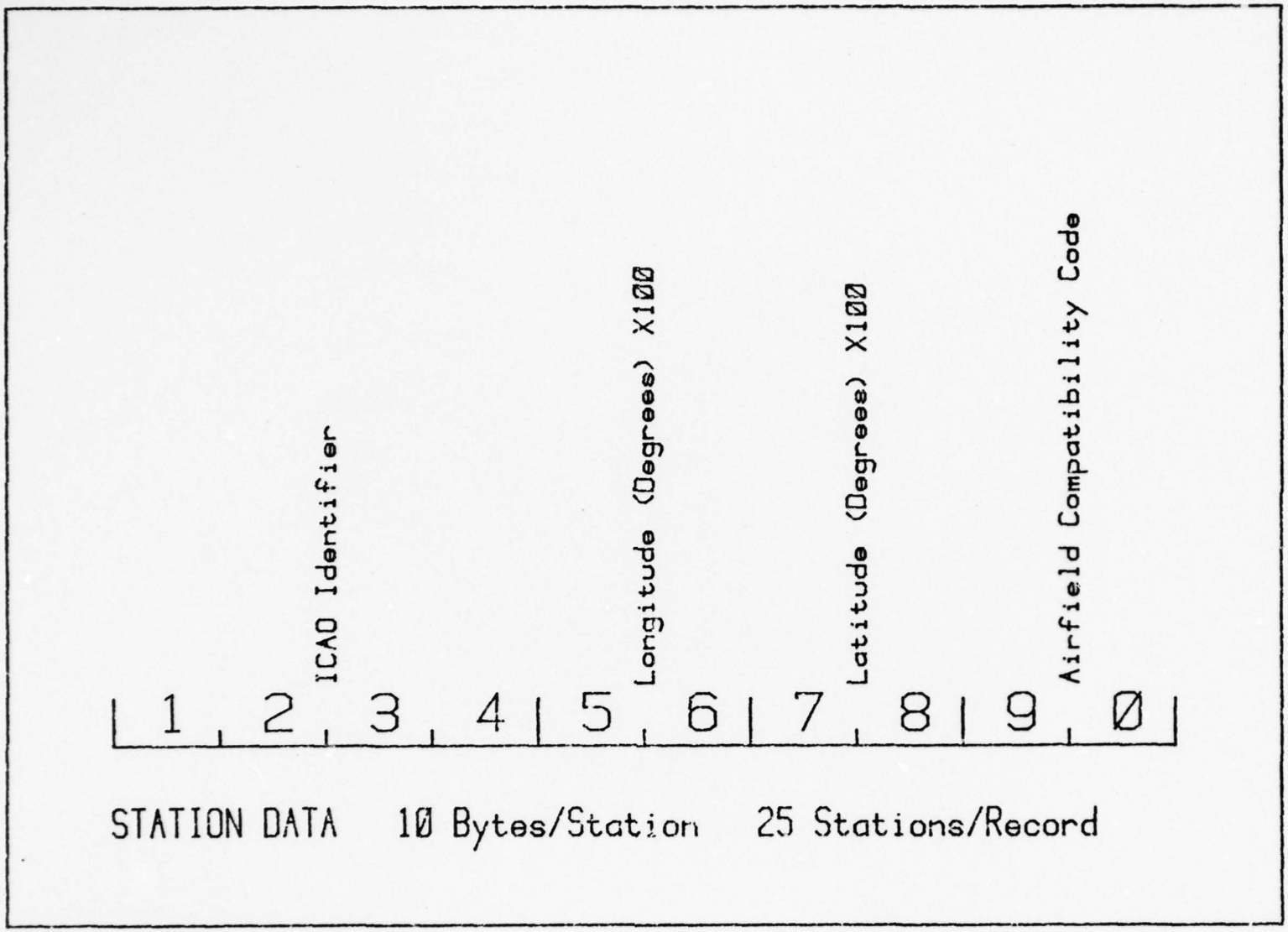


Fig L-5. Data Format for "icao" File Records

APPENDIX M

USER'S GUIDE

APPENDIX M

USER'S GUIDE

The Opportune Rescheduling System provides a means to accomplish the following independent activites:

a. Select diversion-eligible missions from the Airlift Management Planning System (AMPS) data base, and transfer of the mission data to disk storage in the Hewlett-Packard system.

b. Store data on airfield locations and capabilities, political and land-mass boundaries for maps, and airlift mission, requirement, and mission diversion information.

c. Add, change or delete stored data.

d. Selectively print or plot stored data.

e. Analyze all data to select the best possible options for diversion to a given airlift requirement.

f. Selectively print or plot mission diversion options.

To accomplish these activities, the system is divided into four basic modes of operation. The TRANSFER mode requires the system operator to perform several independent tasks to get diversion-eligible mission data stored on the Hewlett-Packard disk. The ENTER mode provides the capability for the user to add, change, or delete data stored in the system on map boundaries, airfields, airlift missions, and airlift requirements. The OPTIMIZE mode performs the mission analysis to select diversion options for a given airlift requirement. The DISPLAY mode provides the capability to print system data, or to plot the data on a map.

While the TRANSFER mode requires several independent tasks, the other three modes are operated and controlled from the Hewlett-Packard calculator keyboard. Prompt messages shown on the calculator display guide the user in controlling the system to accomplish the desired activities. Procedures for use of the four modes of operation will be explained in the following sections.

TRANSFER MODE PROCEDURES

The WWMCCS system INQUEST program to select and store diversion-eligible missions is stored in file DOOMA/FAT/AMCC. Instructions to set the dates for which missions will be selected are included with the program; when these have been set, execute the AMCC program. The program, when executed, stores diversion-eligible mission data in the DODMA/FAT/HPSCED file.

To transfer data from the HPSCED file on WWMCCS to the "msns" file on the Hewlett-Packard disk, disconnect the teletype from its modem. Using the modem extension cord and the RS232 patch cord, connect the modem to the signal converter (188 connection). Remove the plotter interface cord from the HP calculator, and insert the HP98036A Option 1 interface card; connect the HP98036A Option 1 RS232C plug to the signal converter (E1A connection). Insure the signal converter is connected to the calculator, and that the OPPORTUNE RESCHEDULING disk is loaded in the disk drive. These physical connections allow communications between the WWMCCS system and the Hewlett-Packard system. By executing the OPPORTUNE RESCHEDULING System "trans" program, the calculator keyboard

may be used in practically the same manner as the teletype keyboard. To effect data transfer, perform the following at the calculator keyboard:

1. Key in: get "TRANS"; EXECUTE Run

Prompt Message: CRT connected? 1 = Yes; 0 = No.

If the CRT is connected, WWMCCS messages will be displayed in the CRT; otherwise the messages will be printed on the internal printer.

2. Key in: either 1 or Ø as appropriate

Prompt Message: INITIALIZE MISSION FILES 999 = Yes

Initializing the mission files erases all previous data in the file. New data will overwrite old data until the data transfer is complete, but if there is more old data than new, some of the old data will remain in the file if it is not initialized. Any entry other than "999" will not erase old data; if the entry is between 1 and 200, new data will be written on the mission file beginning with that record number. If a data format error occurs, an error message will give the record number where the error occurred. By executing the program again, and entering that record number, the data transfer may continue from the point where the error occurred.

3. Key in: 999 or record number or Ø as appropriate.

999 initializes the mission file, stores new data at beginning of file.

A record number entry does not initialize the file, and stores new data starting with that record number.

Ø does not initialize the file, and stores new data at the beginning of the file.

4. When the calculator display is clear, the calculator keyboard performs the same functions as the teletype keyboard, except:

The STORE key is the equivalent of the teletype CARRIAGE RETURN.

The EXECUTE key is equivalent to the teletype Line Feed.

The F_1 key is equivalent to the teletype CONTROL.

The F_0 key is equivalent to the teletype BREAK.

5. Sign on to WWMCCS time-sharing system, using the same procedures as for the teletype. Time-sharing prompt messages will be displayed on the CRT or the internal printer. Time-sharing messages which are not terminated with a carriage return are not displayed automatically. The carriage return is a signal to the transfer program that a message is complete and can be displayed. If an expected time-sharing message is not displayed, key in: F_1 π. Data or messages waiting for a termination signal will be displayed after this key sequence.

6. When signed onto time sharing:

WWMCCS Message: SYSTEM (F_1 π needed to display this message).

Key in: CARD

WWMCCS Message: OLD or NEW (F_1 π needed to display this message).

Key in: NEW

WWMCCS Message: READY

Key in: BCDA DOOMA/FAT/HPSCED

WWMCCS Message: LINE NUMBERS? (F_1 π needed to display this message).

Key in: ASIS

WWMCCS Message: TAB CHARACTERS AND SETTINGS? (F_1 π needed to display this message).

Key in: STORE Key (Carriage return)

WWMCCS Message: * (F_1 π needed to display this message).

Key in: LIST

7. At this point, WWMCCS will begin sending mission data; the calculator program will recognize the data format, display each data record on the CRT, if connected, and store the data in the "msns" disk file. The process normally requires one to two hours. When the transfer is complete, log off the time-sharing system and return the Hewlett-Packard equipment and the teletype to their normal configuration.

8. If a data format error occurs, a record number will be printed with the error message on the internal printer, as well as an error code on the calculator display. Run the transfer program again; key in the record number on the INITIALIZE MISSION FILE message. The transfer will continue storing data transmitted by WWMCCS at the file record where the error occurred, and all previously stored data will not be lost.

ENTER, OPTIMIZE, AND DISPLAY MODE PROCEDURES

To operate in either of these three modes, check that the plotter, printer, disk and CRT interface cords are connected to the calculator, and that the OPPORTUNE RESCHEDULING disk is in the disk drive. Key in: get "control" and press the EXECUTE key to load the rescheduling system's executive control program. Press the RUN key to start the program. Prompt messages will appear in the calculator display which offer the user options regarding what he wants the system to do next, or tell the user what data must be entered next to accomplish a previously-selected function. Sometimes a prompt message will identify a number to be entered in order to select a certain option; at other times, the first two letters of the

option name are capitalized to indicate that the two capital letters should be entered to select that option. For either option selections or data entries, key in the entry and press the CONTINUE key to restart the program.

When the executive control program is run, the following prompt message is displayed:

Prompt Message: START MAP OVERLAY NOW? YES/NO

A YES entry initiates prompt messages to determine the map boundaries and pen colors to be used in plotting the mpa. See the DISPLAY mode procedures for an explanation of the entries to set map boundaries, etc. When the data on boundaries and pen colors has been entered, the system will complete the map plot before other options are presented.

A NO entry, or merely pressing the CONTINUE key, bypasses the immediate map plot. A map plot may be selected at any time during system operation in the DISPLAY mode.

Prompt Message: FUNCTION: OPTIMIZE, ENTER, DISPLAY

The user's entry at this point selects the mode of operation desired. Also, the program always returns to this point when the user signifies that he is finished with a particular mode of operation. To select an option, enter the two capitalized letters and press the CONTINUE key. The following sections describe each of the three modes in detail.

ENTER Mode Procedures

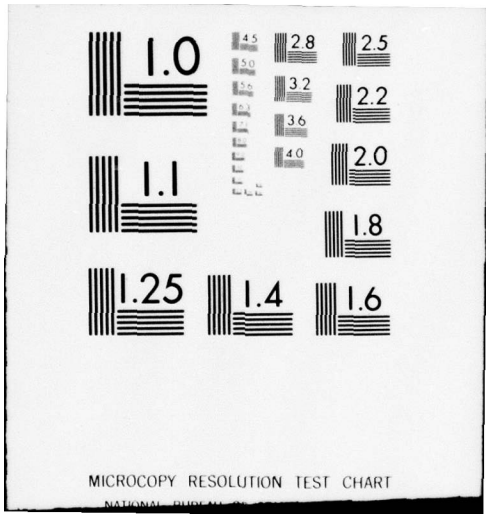
Prompt Message: MSns, REqmts, ICao, COords, SPec, DOne

AD-A064 065 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 15/5
OPPORTUNE RESCHEDULING SYSTEM FOR MILITARY AIRLIFT COMMAND CARG--ETC(U)
DEC 78 G F SANDERSON

UNCLASSIFIED AFIT/GCS/EE/78-17 NL

3 OF 3
AD
A064065

END
DATE
FILMED
3-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART

Entering DO returns the system to the FUNCTION: OPtimize, ENter, DIsplay prompt message. Selecting any of the other data sources will give the user the option to add, change or delete that type of data. MS refers to diversion-eligible mission data; RE is requirement data; IC is airfield data; CO is map coordinate data, and SP is special mission data stored for display purposes.

Prompt Message: ADd, CHange, DElete, or QUit.

A QU entry returns the system to the previous prompt message to select a data type. An AD entry produces a series of prompt messages to guide the user in entering the individual elements of data which comprise the data type selected. For example, if airfield data has been selected, the following prompt messages appear: ICAO Identifier

LONGITUDE

LATITUDE

Suitability Codes C5/C141/C130

An explanation and definition of these data elements is included at the end of this section.

A CH entry produces a prompt message requesting the user identify the specific mission, requirement, airfield, map coordinate or special mission to be changed. Mission and special mission data are identified by mission sequence number or subsequence number, where the sequence number identifies the mission, and the subsequence number identifies the specific mission leg within that mission. Requirements are identified by the requirement number (a value from one to nine) assigned to them when they are entered.

Airfields are identified by the four-letter ICAO identifier. Map coordinates are identified by the record number and index corresponding to how they have been stored on the disk file. When coordinates are added to the map coordinate file, they are stored after the last record and index already in use; the Display mode, explained in the next section, can be used to find record and index numbers for coordinates which need to be changed.

When the specific data to be changed has been identified, it will be displayed on the CRT, and a prompt message similar to the following will be shown on the calculator display.

Prompt Message: Right Data = 1, Try Again = 2, Done = Ø

If a 2 is entered, the system returns to the point where the user is asked to identify the data to be changed. If a Ø is entered, the system returns to the prompt message for selecting a type of data. If a 1 is entered, a series of prompt messages, identifying individual data elements to be changed, are produced. If there is no change to a particular data element, pressing the CONTINUE key leaves the existing data for that element unchanged, and the next prompt message is issued. If the data for that element is incorrect, enter the correct data and press the CONTINUE key. When all of the data has been entered, the new data will be displayed on the CRT, and the following prompt message will be shown on the calculator display.

Prompt Message: CORRECT/STORE = 1, Re-enter Data = 2, Quit = Ø

A 1 entry stores the new data and returns the system to the prompt message for selecting a data type. A Ø entry leaves the old data

unchanged, and returns the system to the prompt message for selecting a data type. A 2 entry starts the series of prompt messages again.

A DE entry to the prompt message ADd, CHange, DElete or QUit produces a prompt message requesting the user to identify the specific data to be deleted. When the data has been identified it will be displayed on the CRT, and a prompt message similar to the following will be shown on the calculator display:

Prompt Message: RIGHT DATA = 1, Try Again = 2, Quit = Ø

If a 2 is entered, the system returns to the point where the user is asked to identify the data to be deleted. If a Ø is entered, the system returns to the prompt message for selecting a type of data. If a 1 is entered, the data is deleted, and the system returns to the prompt message for selecting a data type.

An explanation of each data element for the various types of data follows.

Mission and Special Mission Data

Mission Sequence Number - A number which uniquely identifies one or more mission legs as a scheduled mission. Mission data transferred from WWMCCS are assigned sequence numbers starting from 1.

Mission Subsequence Number - A number which identifies a specific mission leg within a mission. The first scheduled leg of every mission is numbered 1; succeeding legs are numbered sequentially.

Mission ID - The 12-digit MAC mission identifier.

Type Aircraft - C5, C141 or C130. Other entries will not be recognized by system parameters for optimization or display.

Dpearture ICAO - 4-letter ICAO identifier of the mission leg departure station.

ETD - Estimated Time of Departure; 4 digit, Zulu time.

Departure Julian Day - 3-digit julian day for departure date.

Arrival ICAO - 4-digit ICAO identifier for the arrival station.

ETA - Estimated Time of Arrival, 4-digits, Zulu time.

Arrival Julian Day - 3-digit julian day for arrival date.

Cargo (tons) - Tons of cargo scheduled on the mission leg.

Pax - Number of passengers scheduled on the mission leg. Cargo and pax data are informative data; in its present configuration, it is not used by the system since WWMCCS data on scheduled cargo and passengers is unreliable.

REQUIREMENT DATA

REQ NUMBER - A number between one and nine which will be used by the system to identify and locate the specific requirement information. Data on up to nine airlift requirements may be stored.

ONLOAD ICAO - 4-digit ICAO identifier of the airfield where the airlift requirement originates.

ONLOAD NOT-EARLIER-THAN TIME - 4-digit Zulu time when the cargo or passengers first become available for pick-up.

ONLOAD NET DATE - 3-digit julian day for the NET (above) date.

OFFLOAD ICAO - 4-digit ICAO identifier of the airfield where the cargo and/or passengers must be delivered.

OFFLOAD NOT-LATER-THAN TIME - 4-digit Zulu time by which the cargo and/or passengers must be delivered.

OFFLOAD NLT Date - 3-digit julian day for the NLT (above) date.

CARGO (Tons) - Tons cargo involved in this airlift requirement.

PAX - Number of passengers involved in this airlift requirement. Cargo and passenger data are informative only; in its present configuration, the system does not use this data in the optimization analysis.

CARGO SUITABILITY C5/C141/C130 - A three-digit code which is used by the system to determine which types of aircraft could airlift the requirement. The left-most digit corresponds to the C5; the middle digit to the C141, and the right-most digit to the C130. A 1 in any position signifies that the cargo and/or passengers may be carried on that type aircraft; a 0 in any position signifies that the requirement is incompatible with that type aircraft. For example, a 110 means the requirement may be airlifted on either a C5 or C141, but cannot be scheduled on a C130.

AIRFIELD DATA

ICAO Identifier - 4-digit ICAO identifier letters for the airfield.

LONGITUDE - Longitude in degrees/tenths/hundredths for the airfield. Longitudes in the Western hemisphere must be entered as negative. For example: -105.50 corresponds to longitude 105 degrees 30 minutes West.

LATITUDE - Latitude in degrees/tenths/hundredths for the airfield. Latitudes in the Southern hemisphere must be entered as negative.

SUITABILITY CODES C5/C141/C130 - A three-digit code used by the system to determine which types of aircraft may safely land at the airfield.

The left-most digit corresponds to the C5; the middle digit to the C141; and the right-most digit to the C130. A 1 in any position signifies that the airfield is suitable for that type aircraft; a Ø signifies that that type aircraft cannot land at the airfield.

MAP COORDINATES

Map coordinate data is stored as geographical coordinates for individual points, plus a code which indicates what kind of line the plotter must draw as it moves to that point. The plotter moves from point to point in the same sequence that the coordinates are stored. A code of 1 directs the plotter to move to the longitude and latitude specified without drawing a line; a code of 2 directs the plotter to draw a straight line from its previous position to the longitude and latitude specified. A code number greater than two directs the plotter to draw a counterclockwise arc, with a radius in nautical miles equal to the code value, from the plotter's previous position to the coordinates specified. A negative code uses the same convention, except the arc is drawn in the clockwise direction. A code value of zero is ignored by the plotter; it disregards any coordinates corresponding to that code, and remains at its previous position.

In entering coordinates to define a map plot, the user must determine the sequence of points he wishes the plotter to trace, and the type of line (straight or arc) to be drawn.

LONGITUDE - Longitude in degrees/tenths/hundredths for the point to be plotted. Longitudes in the Western hemisphere must be entered as negative.

LATITUDE - Latitude in degrees/tenths/hundredths for the point to be plotted. Latitudes in the Southern hemisphere must be entered as negative.

CODE - Plotter code values as explained above. Code values must be whole numbers; decimals are ignored.

The next section describes procedures for the OPTIMIZE mode of operation.

OPTIMIZE Mode Procedures

When the OPTIMIZE mode is selected, the following prompt message is displayed:

Prompt Message: NEW OPTIMIZE FACTORS? 1 = YES; Ø = NO

Optimize factors are used in the selection of diversion options. The factors include the following parameters for each type aircraft: flying hour costs, extra flying time for take-offs and landings, rescheduling weighting factors, and aircraft block speeds. Flying hour costs are the computed dollar costs necessary to operate an aircraft for one hour of flying; they allow comparisons between different types of aircraft to select the most economical options. Flying time per stop is the extra flying time needed by an aircraft for each take-off and landing, irrespective of the route to be flown. Rescheduling weighting factors allow either an advantage or disadvantage to be given to an aircraft type in the mission analysis and selection process. A weight factor of 1 is neutral; a factor between Ø and 1 gives

that type aircraft an advantage in the selection process. The closer the value is to Ø, the greater the relative advantage. A factor greater than 1 gives that type aircraft a disadvantage; the greater the number, the greater the disadvantage. For example, if the C5's are overflying their monthly flying hour commitment, a factor of 2 would mean that a C5 mission must be twice as economical as missions flown by other aircraft types to be selected as a diversion option. Aircraft block speeds are the average speed, in knots, for that type aircraft over a typical mission route.

If NEW OPTIMIZE FACTORS are selected, the system prints the default values, or the last values entered, and then issues prompt messages for each of the factors and aircraft types.

Prompt Message: REQ # (Enter Ø to Input New Req)

If the requirement data has already been stored, entering the requirement number will cause the data to be displayed on the CRT, and the mission analysis and selection of diversion options will proceed using that data as a basis. If the requirement data must be entered, a Ø input will produce a series of prompt messages identify the data elements to be entered. (See ENTER mode procedures for requirement data.) When the new data has been entered, it will be displayed on the CRT, and the prompt message RIGHT REQMT 1 = YES, 2 = NO, Ø = QUIT will be shown. A 1 entry causes the mission analysis and selection of diversion options to proceed;

a 2 entry returns the system to the REQ # (Enter Ø to Input NEW REQ) prompt message. A Ø entry returns the system to the FUNCTION: OPtimize, ENter, DIsplay prompt message.

Mission analysis and selection of diversion options may take up to 40 minutes to complete; while the analysis is taking place, the message "CHECKING MSN RECORD" and a record number is shown on the calculator display. Since the number of disk records used to store the mission data was displayed at the end of the TRANSFER mode of operation, the user is given some idea of where the system is in its analysis of mission data. When the process is completed, data on the diversion options is displayed on the CRT, and the following prompt message is displayed:

Prompt Message: DISPLAY RESULT = 1, RECOMP = 2, DONE = Ø

A Ø, or any entry other than a 1 or 2, returns the system to the FUNCTION: OPtimize, ENter, DIsplay prompt message. The diversion data is stored, however, and may be recalled at any time via the DISPLAY mode. A 2 entry causes the system to reanalyze the mission data file to select the next six-best diversion options for the requirement. Data on the first six options is no longer stored. A 1 entry invokes the DISPLAY mode, and allows the user to select a plotted or printed data presentation. For an explanation of this capability, see the procedures for the DISPLAY mode (OP display) in the next section.

DISPLAY Mode Procedures

When the DISPLAY mode is selected, the following prompt message is shown on the calculator display:

Prompt Message: MS, MI, RE, CO, IC, SP, PE, MA, OP, DOne

MS refers to mission data; all mission legs of a mission are displayed. MI refers to data on individual mission legs. RE is requirement data; CO is map coordinates; IC is airfield data, SP is special mission data; PE is used to re-designate the pen colors used to plot the various data types; MA is used to set new map boundaries for a map plot; and OP refers to the collection of mission, requirement and diversion option data presented in a special format.

A DO entry returns the system to the FUNCTION: OPtimize, ENter, DIsplay prompt message. An MA entry causes a series of prompt messages which guide the user in selecting map boundaries, a longitude/latitude grid overlay on the map, and the color of pens to be used in plotting the map, missions, requirements, etc. A PE entry allows the pen color selections to be made independently of the map boundary selections. For an explanation of the data entries to select map boundaries, map grid, and pen colors, see the end of this section.

The remaining entries all select data for display. After the data has been selected, prompt messages to indicate the type of display desired will be shown in the calculator display.

Prompt Message: PLOT? 1 = YES, Ø = NO

A 1 entry ultimately produces a plot of the data selected.

A Ø entry bypasses the plot option.

Prompt Message: PRINT? 1 = YES, Ø = NO

A 1 entry ultimately causes the data selected to be printed on the Hewlett-Packard printer. A Ø entry results in the data being displayed on the CRT.

At this point, the specific mission requirement, airfield, etc., to be plotted or printed must be identified. Since the system responds differently for each type of data, each will be explained in the following subsections.

MS (Mission Data)

Prompt Msg: MSN SEQUENCE NBR

The sequence number for the mission, including all its mission legs, to be plotted or printed must be entered. For an explanation of the sequence number, see the mission data description in the procedures for the ENTER mode of operation. When the sequence number is entered, the data stored on all the mission legs of that mission is displayed on the CRT, and the prompt message RIGHT MSN = 1, TRY AGAIN = 2, QUIT = Ø is shown in the calculator display. A 1 entry causes the system to plot or print the mission data according to the previous selections for those options. A 2 entry returns the system to the MSN SEQUENCE NBR prompt message; a Ø entry returns the system to the MS, MI, RE, CO, IC, SP, PE, MA, OP, DOne prompt message.

MI (Mission Leg Data)

The system operates the same as for mission data, except that the subsequence number for the specific mission leg must also be entered. For an explanation of subsequence numbers, see the mission data description in the procedures for the ENTER mode of operation. Only the specific mission leg identified will be plotted or printed.

SP (Special Mission Data)

The system operates the same as for mission leg data, except that the mission legs identified are only those stored in the special mission file. Only 6 mission legs may be stored in the special mission file.

RE (Requirement Data)

Prompt Message: REQ # TO BE DISPLAYED

The requirement number (from 1 to 9) must be entered. Data for that requirement is displayed on the CRT. As with the mission data, the system requests the user to confirm that the correct data to be plotted or printed has been identified by issuing the prompt message RIGHT REQMT = 1, TRY AGAIN = 2, QUIT = Ø. Responses are the same as for the mission data.

IC (Airfield Data)

Prompt Message: ICAO to be displayed

The four-letter ICAO identifier for the airfield to be displayed should be entered. If data on the airfield is stored, it will be displayed on the CRT, and plotted or printed as previously selected. If there is no data stored for the airfield, an error message is printed on the calculator internal printer, and the system returns to the data selection prompt message.

CO (Map Coordinates Data)

Prompt Message: SHOW COORDINATES ON CRT? YES = 1

Individual map coordinates may not be plotted or printed; rather a Full disk record, of up to 42 coordinates, is selected for plotting or printing. The selection is made by entering the record number of the coordinates to be displayed. The above prompt message allows each set of coordinates, and their corresponding plotter code, to be displayed on the CRT as they are being plotted or printed. For an explanation of map coordinates and the plotter code, see the map coordinates data description in the procedures for the ENTER mode of operation.

If the CRT display is not desired, press the CONTINUE key.

Prompt Message: STARTING REC #...CONTINUE FOR FULL MAP

The map coordinates display program will plot and/or print all coordinates stored on the disk records beginning with the starting record number through the ending record number. Enter the lowest record number containing the coordinates to be displayed. If a complete map, or a print of all map coordinates is desired, press the CONTINUE key.

Prompt Message: ENDING REC #...CONTINUE FOR A FULL MAP

Enter the highest record number containing coordinates to be displayed. If only one record of coordinate data is to be displayed, enter the same record number for both the starting and ending record number. If a complete map, or a print of all map coordinates is desired, press the CONTINUE key. When the ending record number has been entered, the system responds by plotting and/or printing the map coordinates selected, and then returns to the data selection prompt message.

OP (Optimized Diversion Options)

This selection will plot or print all the data associated with an airlift requirement, missions selected for possible diversion to this requirement, and data on the diversion routing, extra flying hours and dollar costs incurred, etc. For such a display to be presented, the OPTIMIZE mode must have been executed against the airlift requirement, and all of the mission, requirement and diversion data must still be stored. If any of the data has been changed (for example, if new mission data has been transferred from WWMCCS), the system will plot or print the new data, but there may be no relationship between the requirement, missions and diversion information. When the diversion data is created by the OPTIMIZE mode of operation, it stores the requirement number and mission sequence numbers selected as diversion options. If requirement or mission data is changed after the diversion data is created, the relationship is lost.

The presentation is identified by the requirement number. When the user selects the OP display, the following prompt message is shown in the calculator display:

Prompt Message: REQUIREMENT NBR

Enter the requirement number for the display desired.

Prompt Message: FIRST OPTION TO BE PRESENTED

The system computes and stores six diversion options for an airlift requirement. They are numbered from one to six according to their relative costs for diversion. The system will plot or print data on the diversion options beginning with the number of the first option selected through the last option selected.

Prompt Message: LAST OPTION TO BE PRESENTED

When the first and last diversion option numbers have been entered, data on the previously selected requirement is presented on the CRT, and the following prompt message is shown on the calculator display:

Prompt Message: RIGHT REQMT? 1 = YES, 2 = NO, Ø = QUIT

A Ø entry returns the system to the data selection prompt message. A 2 entry returns the system to the REQUIREMENT NBR prompt message. A 1 entry causes the selected data to be plotted or printed, after which the system returns to the data selection prompt message.

MA (Select Map Boundaries)

The MA entry allows the user to select map boundaries, pen colors of plotted data, and a grid overlay for plotted data. Since the system plots all data with reference to the last map boundaries selected, the same boundaries should be maintained for all data presented on a single plotting paper. If the MA option is not selected before a plotting display is called for, the system uses a default set of map boundaries which will plot all data falling within the continental US. Selecting the MA option implies that a new plot is being initiated, and the following prompt message is displayed:

Prompt Message: CONTINUE when Plotter Ready

The system is halted to allow the user to place fresh paper on the plotter, and to set the mechanical plotting limits P1 and P2 (see the HP9872A Graphics Plotter Manual for instructions). When this is accomplished, press the CONTINUE key.

Prompt Message: MAP EAST BOUNDARY

Enter the east boundary of the map or data display to be plotted in degrees/tenths/hundredths of longitude. For longitudes in the western hemisphere, enter as a negative number. For example 105 degrees 33 minutes west should be entered as -105.55. Pressing the CONTINUE key without entering a longitude selects -67.Ø, the default for the east boundary.

Prompt Message: MAP WEST BOUNDARY

Using the same convention as for the east boundary, enter the west boundary for the map or data display to be plotted. Pressing the CONTINUE key without entering a longitude selects -125.00 as the west boundary.

Prompt Message: MAP NORTH-SOUTH MIDPOINT

Enter the latitude of the center of the map or data display to be plotted in degrees/tenths/hundredths. For latitudes in the southern hemisphere, enter as a negative. It is necessary to enter the latitude of the center of the plot, rather than the northern and southern boundaries, since the limits on the plotting paper have already been established by the eastern and western boundaries, and by the mechanical limits set by P1 and P2.

Prompt Message: NEW PEN COLORS 1 = YES, Ø = NO

A 1 entry allows selection of pen numbers, corresponding to the numbers for each pen color on the plotter pen rack, which will be used to plot map coordinates, airfields, missions, requirements, diversions, and grid overlay. A Ø entry bypasses the option, and the default color conventions are used. See the explanation for the PE entry in the next subsection.

Prompt Message: MAP GRID SIZE; NO GRID...CONTINUE

A numerical entry causes a longitude-latitude grid to be plotted within the map boundaries selected. The size of each grid square in degrees will be the number entered. For example, a 10 entry causes 10 degree grid squares to be plotted. Longitudes and latitudes of 10 degrees, 20 degrees, 30 degrees, etc., will be shown where they fall within the map boundaries selected. Pressing only the CONTINUE key bypasses the grid plot option.

Prompt Message: MAP BORDER? 1 = YES

A 1 entry causes a frame to be plotted around the plotting area on the paper. Any other entry bypasses this option. After this entry, the system returns to the data selection prompt message.

PE (Select Plotting Pen Colors)

The system can plot each type of data (missions, airfields, map coordinates, map grid, requirements, diversions) in up to 4 different colors. The PE entry allows a selection of the colors to be used for each. When PE is entered, the currently designated pen numbers, corresponding to the pen colors on the plotter pen rack, are printed on the calculator internal printer for each type of data. A series of prompt messages are shown to get new pen number designations. The default values are as follows:

Green: Map Coordinates, Map Grid, Airfields

Red: Diversions

Blue: Missions

Black: Requirements

When new pen number designations have been entered, they are printed on the calculator internal printer, and the system returns to the data selection prompt message.

ERROR Messages

The system evaluates entries input by the user, and will print an error message if the entry is unacceptable. In most cases, the error message is printed, and the system returns to the prompt message where the unacceptable entry occurred. In some cases, the entry may be acceptable, but the requested operation cannot be performed. (For example, a request to plot an airfield for which no data is stored.) In such cases, an error message will describe the problem, and the system will return to the point at which the operation was requested.

Since there are a large number of possible errors and error messages, they will not be listed in detail. Error messages are sufficiently descriptive of the error that the user should be able to correct his entry after referring to User's Guide procedures.

If the calculator program halts, and a calculator error message is shown in the calculator display, a system program error is indicated. Copy the error code, and the operation being performed when the error occurred, and advise the MAC/DOOMO branch chief who will arrange for the necessary program corrections.

APPENDIX N

SAMPLE DATA DISPLAYS

APPENDIX N

SAMPLE DATA DISPLAYS

The Opportune Rescheduling System has the capability for printing or plotting any of the information stored in the data files. This Appendix contains samples of some of the display capabilities.

Data displays in text format are directed to the Hewlett-Packard CRT unless a hard-copy print has been requested. The output format is the same in either case.

Fig N-1 shows a geographical plot of map coordinates and ICAO-coded airfield locations. Fig N-2 is a sample of mission data in printed form; Fig N-3 is a plot of the same data. Fig N-3 also shows the selectable map boundary capabilities. Fig N-4 is a sample of a diversion options presentation in printed form; Fig N-5 is a plot of one of the diversions. The actual plotted information is normally produced in four colors to make the display more readily understandable.

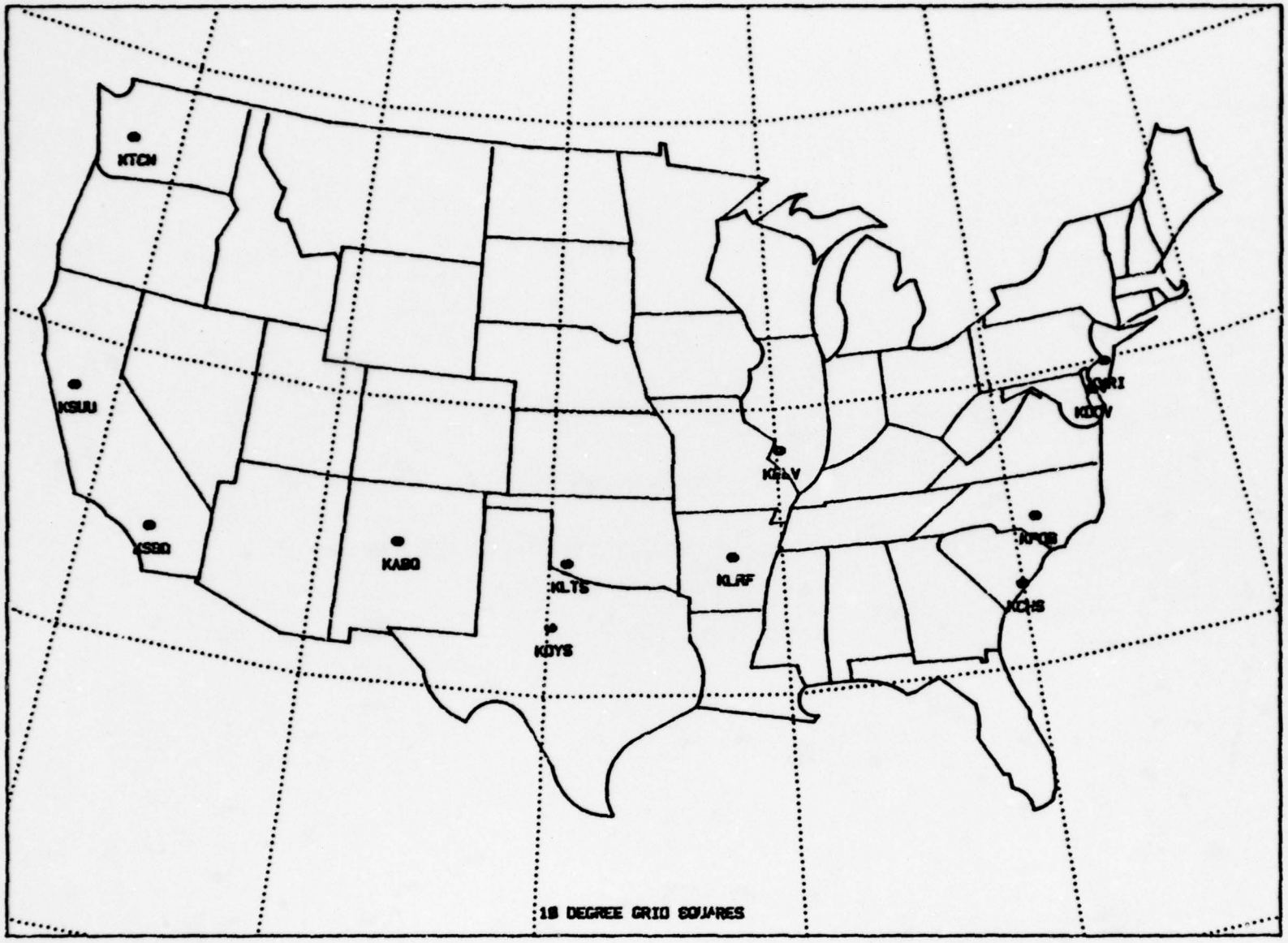


Figure N-1. Sample Plot of Map Coordinates and Selected Airfields.

SEQ	NO	MISSION ID	TYPE	DEPT	DAY	ETD	ARR	DAY	ETA	CGO	PAX	HRS
50	1	AJM173905271	141	KCHS	271	2200	KDOV	271	2345	8224	8224	1.8
50	2	AJM173905271	141	KDOV	272	300	EDAR	272	1130	8224	8224	8.5
50	3	AJM173905271	141	EDAR	273	700	EDNJ	273	800	8224	8224	1.0
50	4	AAM173905271	141	EDNJ	273	1015	EGUN	273	1130	8224	8224	1.3
50	5	AAM173905271	141	EGUN	273	1445	KWRI	273	2345	8224	8224	9.0
50	6	AAM173905271	141	KWRI	274	300	KBLV	274	500	8224	8224	2.0
50	7	AAM173905271	141	KBLV	274	815	KCHS	274	1015	8224	8224	2.0

Figure N-2. Sample of Printed Mission Data

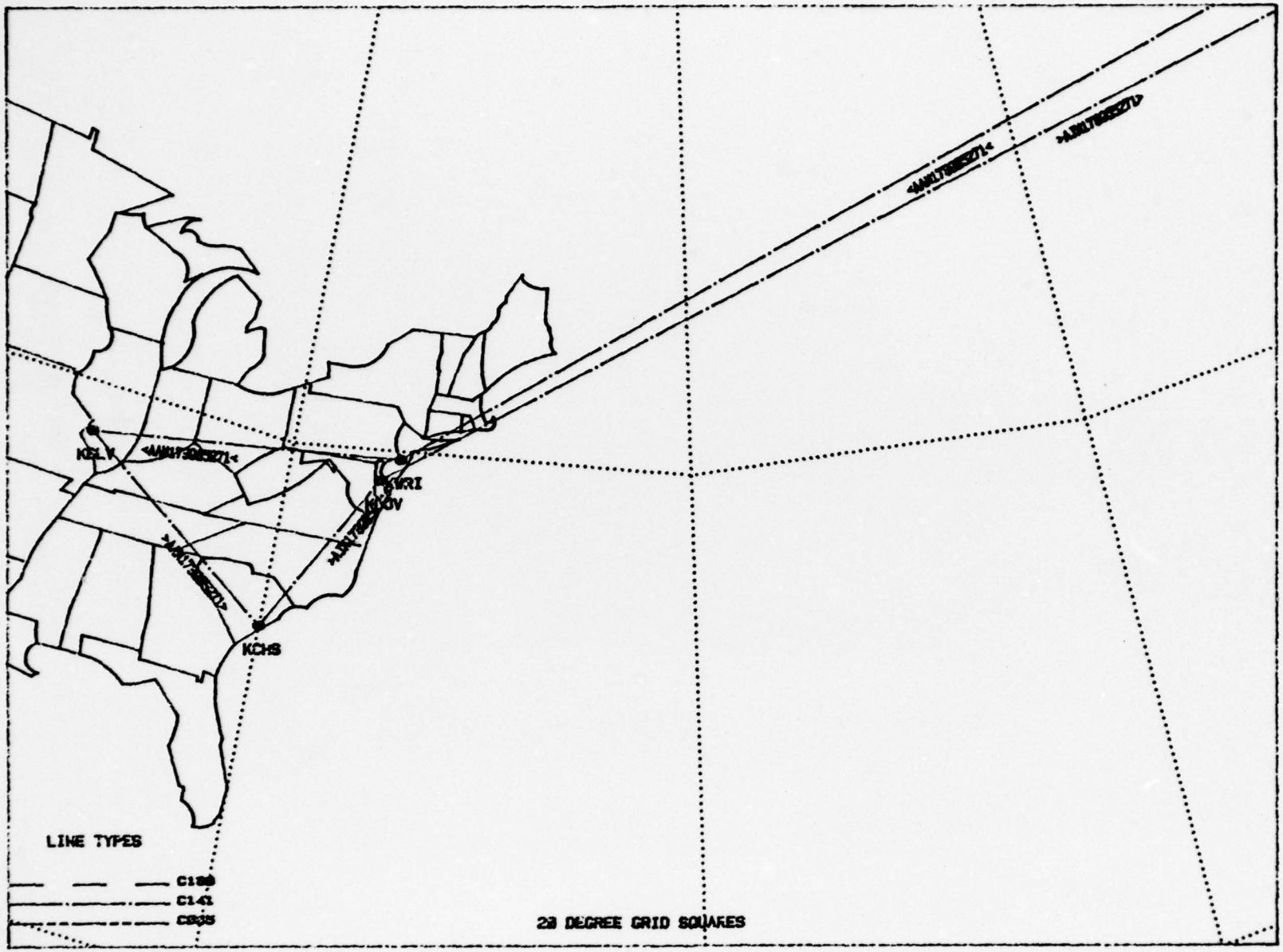


Figure N-3. Sample Plot of Mission Data

SEQ	ONLOAD	NET DAY	OFFLOAD	NLT DAY	CGO	PAX	CODE	N.M.
9	KPBG	100 271	KCVS	2100 271	10	0	11	1505

DIVERSION OPTION #1

SEQ	NO	MISSION ID	TYPE	DEPT	DAY	ETD	ARR	DAY	ETA	CGO	PAX	HRS
131	1	HVA047400268	130	KNGU	269	2145	KDYS	270	230	0	0	4.8

COMP VALUE	EXTRA COST	TYPE A/C	DEP ICAO	FLY HRS	ONL ICAO	FLY HRS	OFF ICAO	FLY HRS	NEXT ICAO	MSN SEQ	SUB SEQ	REQ NO.	TOT HRS
1522	$ 3043	130	KNGU	2.1→	KPBG	5.8→	KCVS	1.1→	KDYS	131	1	9	9.0

DIVERSION OPTION #2

SEQ	NO	MISSION ID	TYPE	DEPT	DAY	ETD	ARR	DAY	ETA	CGO	PAX	HRS
140	1	HJG2075M2270	130	KLRF	270	2025	KPOB	270	2250	0	0	2.4
140	2	HVG2075M2270	130	KPOB	271	345	KLRF	271	615	0	0	2.5

COMP VALUE	EXTRA COST	TYPE A/C	DEP ICAO	FLY HRS	ONL ICAO	FLY HRS	OFF ICAO	FLY HRS	NEXT ICAO	MSN SEQ	SUB SEQ	REQ NO.	TOT HRS
2932	$ 5863	130	KPOB	2.6→	KPBG	5.8→	KCVS	2.4→	KLRF	140	2	9	10.8

DIVERSION OPTION #3

SEQ	NO	MISSION ID	TYPE	DEPT	DAY	ETD	ARR	DAY	ETA	CGO	PAX	HRS
139	1	HVG2065K1268	130	KPOB	269	2330	KLRF	270	200	3	8	2.5

COMP VALUE	EXTRA COST	TYPE A/C	DEP ICAO	FLY HRS	ONL ICAO	FLY HRS	OFF ICAO	FLY HRS	NEXT ICAO	MSN SEQ	SUB SEQ	REQ NO.	TOT HRS
2932	$ 5863	130	KPOB	2.6→	KPBG	5.8→	KCVS	2.4→	KLRF	139	1	9	10.8

Figure N-4. Sample of Printed Diversion Options Presentation

SEQ	ONLOAD	NET DAY	OFFLOAD	NLT DAY	CGO	PAX	CODE	N.M.
9	KPBG	100 271	KCVS	2100 271	10	0	11	1505

DIVERSION OPTION #1

SEQ	NO	MISSION ID	TYPE	DEPT	DAY	ETD	ARR	DAY	ETA	CGO	PAX	HRS
131	1	HVA047400268	130	KNGU	269	2145	KDYS	270	230	0	0	4.8

COMP VALUE	EXTRA COST	TYPE A/C	DEP ICAO	FLY HRS	ONL ICAO	FLY HRS	OFF ICAO	FLY HRS	NEXT ICAO	MSN SEQ	SUB SEQ	REQ NO.	TOT HRS
1522	$ 3043	130	KNGU	2.1→	KPBG	5.8→	KCVS	1.1→	KDYS	131	1	9	9.0

DIVERSION OPTION #2

SEQ	NO	MISSION ID	TYPE	DEPT	DAY	ETD	ARR	DAY	ETA	CGO	PAX	HRS
140	1	HJG2075M2270	130	KLRF	270	2025	KPOB	270	2250	0	0	2.4
140	2	HVG2075M2270	130	KPOB	271	345	KLRF	271	615	0	0	2.5

COMP VALUE	EXTRA COST	TYPE A/C	DEP ICAO	FLY HRS	ONL ICAO	FLY HRS	OFF ICAO	FLY HRS	NEXT ICAO	MSN SEQ	SUB SEQ	REQ NO.	TOT HRS
2932	$ 5863	130	KPOB	2.6→	KPBG	5.8→	KCVS	2.4→	KLRF	140	2	9	10.8

DIVERSION OPTION #3

SEQ	NO	MISSION ID	TYPE	DEPT	DAY	ETD	ARR	DAY	ETA	CGO	PAX	HRS
139	1	HVG2065K1268	130	KPOB	269	2330	KLRF	270	200	3	8	2.5

COMP VALUE	EXTRA COST	TYPE A/C	DEP ICAO	FLY HRS	ONL ICAO	FLY HRS	OFF ICAO	FLY HRS	NEXT ICAO	MSN SEQ	SUB SEQ	REQ NO.	TOT HRS
2932	$ 5863	130	KPOB	2.6→	KPBG	5.8→	KCVS	2.4→	KLRF	139	1	9	10.8

Figure N-4. Sample of Printed Diversion Options Presentation

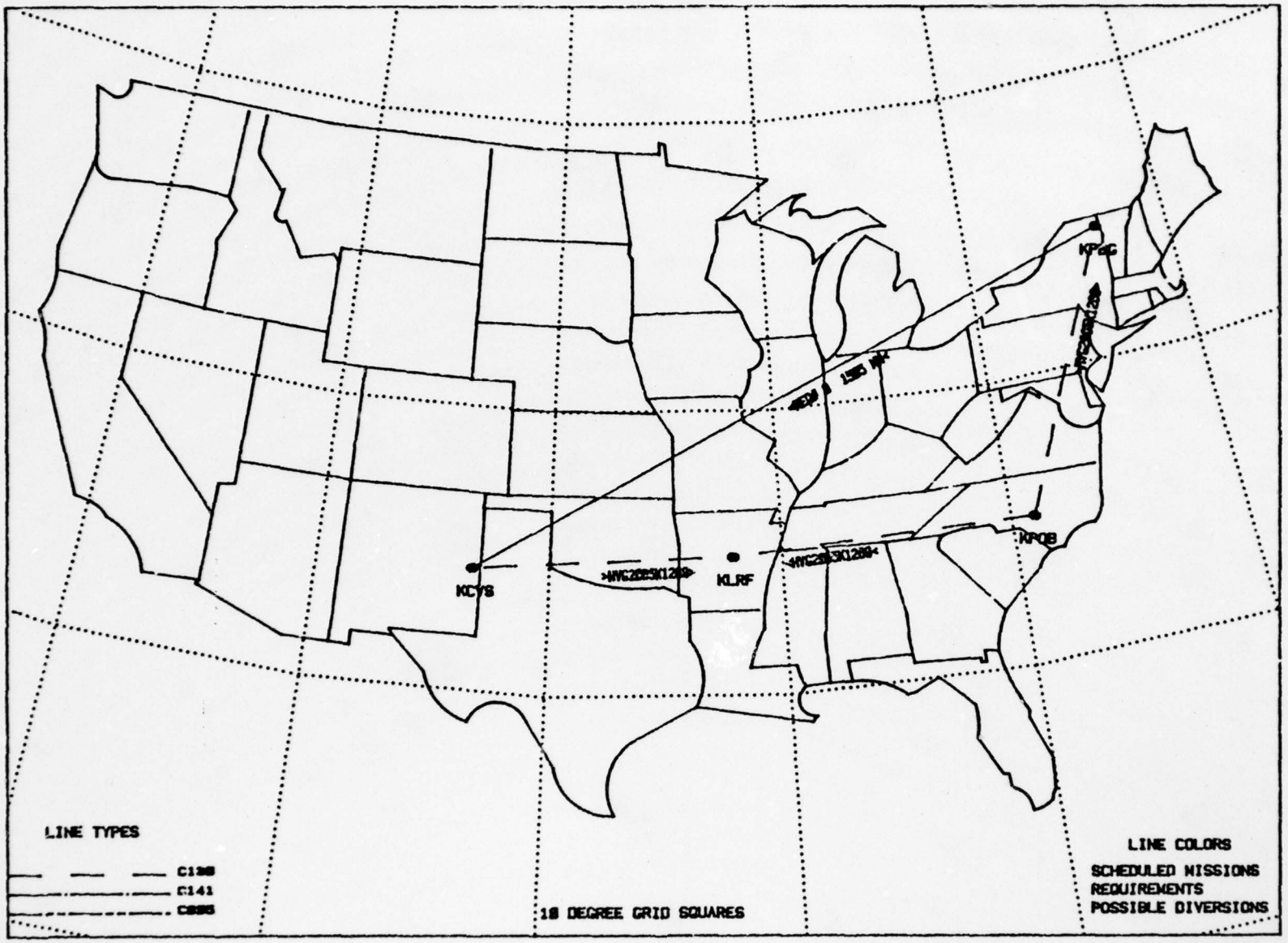


Figure N-5. Sample of Plotted Diversion Option Presentation

VITA

Major Gary Sanderson was born on 17 June 1942 in Louisville, Kentucky. He graduated from Avon Lake High School, Avon Lake, Ohio, in 1960. He attended the United States Air Force Academy, from which he received the Degree of Bachelor of Science, as well as a commission in the United States Air Force, in June 1964. He entered Undergraduate Pilot Training at Randolph Air Force Base, Texas, and completed the training at Stead Air Force Base, Nevada, in August 1965. He served as an HH-43B Rescue Crew Commander at Laredo AFB, Texas, Cam Rahn Bay Air Base, Vietnam, and Incirlik Air Base, Turkey. He served as an HH-53 Rescue Crew Commander, Instructor Pilot and Flight Examiner at Udorn Royal Thai Air Base, Thailand, from July 1969 until July 1970. He completed T-38 jet qualification at Randolph AFB, Texas, and subsequently served as WC-130 Aircraft Commander, Instructor Pilot, and Flight Examiner at Ramey Air Force Base, Puerto Rico and Keesler Air Force Base, Mississippi. From 1975 until 1977, he served as an Air Operations Staff Officer at Headquarters Military Airlift Command, Scott Air Force Base, Illinois. During this period, he earned the degree of Master of Arts from Webster College, St. Louis, Missouri. He entered the School of Engineering, Air Force Institute of Technology in June 1977.

Permanent Address: 629 North Valley Road
Paducah, Kentucky 42001

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GCS/EE/78-17	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) OPPORTUNE RESCHEDULING SYSTEM FOR MILITARY AIRLIFT COMMAND CARGO AND PASSENGER MISSIONS		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Gary F. Sanderson Major USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS HQ Military Airlift Command (MAC/DOOMO) Scott AFB, IL 62225		12. REPORT DATE December, 1978 13. NUMBER OF PAGES 217
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for public release; IAW AFR 190-17
Joseph P. Hipps, Major, USAF
Director of Information 1-23-79

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Airlift
Scheduling

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Military Airlift Command operates numerous C-5, C-141 and C-130 missions daily over worldwide air routes. Many of these flights operate partially full, and represent a considerable airlift capability if their scheduled routes and spare cargo space can be matched with unscheduled cargo or passenger requirements. Using existing data processing equipment at Headquarters Military Airlift

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Command, an automated system was designed and implemented to match these new airlift requirements with scheduled missions, and to compute and display the data necessary for rescheduling decisions. The system transfers scheduled-mission data from a Honeywell H-6080 computer installation to a Hewlett-Packard 9825A Programmable Calculator and associated disk storage. System programs on the HP9825A select the most efficient missions for re-routing to accomplish an additional cargo or passenger requirement, grafically displays the change in routing on maps, and presents printed data on flying hours and monetary costs for each proposed mission diversion. Techniques derived from Structured Analysis and Design Technique and Structured Design methodologies were used to define and structure the system. A major objective of the system design and implementation was to assure the system users would be able to understand and modify any portion of the software to change or improve performance.